IBM MQSeries Workflow

Getting Started with Buildtime

Version 3.2

SH12-6286-03

IBM MQSeries Workflow

Getting Started with Buildtime

Version 3.2

SH12-6286-03

Note!

Before using this information and the product it supports, be sure to read the general information under "Appendix B. Notices" on page 111.

Fourth Edition (June 1999)

This edition applies to version 3, release 2 of IBM MQSeries Workflow (product number 5697-FM3) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SH12–6286–02.

Contents

About this book

This book introduces you to the Buildtime component of IBM MQSeries (R) Workflow, hereafter referred to as MQ Workflow. It describes how you can use Buildtime to create a workflow model. It also introduces you briefly to the modeling tasks, using examples to get an understanding what you can do with Buildtime.

The first part of the book explains how you define your business processes and the resources that you need to run the processes.

The second part describes the Definition Language (FDL), which you can use with MQ Workflow. You can easily import existing workflow definitions into MQ Workflow or export them using the MQ Workflow exchange format FDL.

If you want to explore MQ Workflow in more depth or learn about the technical details, see "Getting help" on page 12.

This book does not contain an overview of all the MQ Workflow components or how to install them. For a list of additional publications that describe other components of MQ Workflow, refer to "MQ Workflow publications" on page 121.

Who should read this book

Read this book if you want to know:

- What you can do with Buildtime
- How you can use Buildtime

If you want to get familar with the concepts of workflow and the architecture of MQ Workflow, see the *IBM MQSeries Workflow: Concepts and Architecture*.

Conventions and terminology used in this book

Convention used	How it is used
Book titles are shown in italics.	*IBM MQSeries Workflow: Concepts and Architecture*
Menu-bar choices and push buttons are shown in bold.	Click **OK**.

Variables are shown in italics. Important information is also shown in *italics*.	The program *program name* is assigned to the activity.
	This symbol flags suggestions, important hints, and practical techniques.

How to get additional information

Visit the MQSeries Workflow home page at
http://www.software.ibm.com/ts/mqseries/workflow

For a list of additional publications, refer to "MQ Workflow publications" on page 121.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other MQSeries Workflow documentation, use one of the following methods:

- Send your comments by e-mail to: swsdid@de.ibm.com

  Be sure to include the name of the book, the part number of the book, the version of MQSeries Workflow, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).
- Fill out one of the forms at the back of this book and return it by mail, by fax, or by giving it to an IBM representative.

Part 1. Modeling with Buildtime

Chapter 1. Introducing Buildtime

With MQ Workflow you can design, refine, document, and control your business processes. MQ Workflow assists you in daily business operations, in planning and management, and also in the design of applications that are tailored to your business. With MQ Workflow you can do the following:

- Define and document your processes
- Run your processes more efficiently:
  - Support the people who are doing the work
  - Fully automate activities that do not require human guidance
  - Administer your workflow

MQ Workflow is a client/server system and Buildtime is the component that you use for defining and documenting your business processes in a workflow model.

A business process typically consists of many activities or even subprocesses that contain more activities. For the various activities in a process, you specify the control flow, the data flow, and the application programs that you want to use within a process.

What is a workflow model?

A workflow model is a complete representation of one or more business processes, comprising all the relevant business activities. It also contains the definitions for the workflow participants and the IT resources you need to accomplish your workflow.

When you define a process, you use dialogs in MQ Workflow Buildtime and a graphical editor to draw process diagrams. To build a workflow model, you need to define the properties for the three main components. Figure 1 on page 4 shows the three main components of a workflow model:

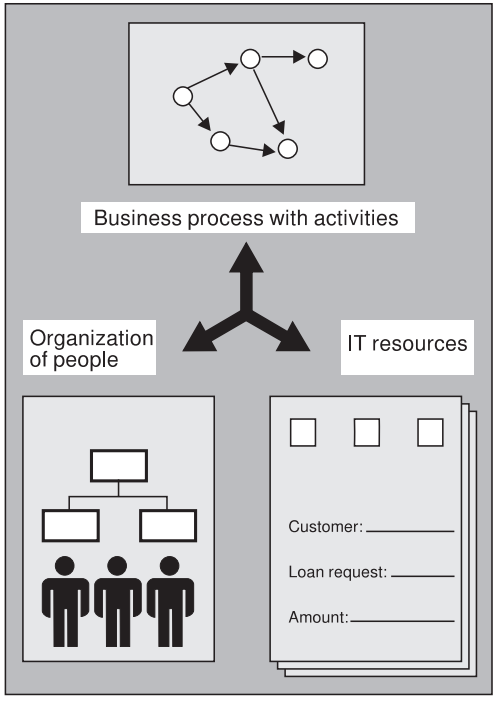


Figure 1. Building a workflow model

- Business process with activities

  With MQ Workflow, you can graphically depict the processes and their activities in a process diagram. You can also define the process logic behind the components of a workflow model.
- Organization of people

  You specify the staff in your organization who must perform the business activities.
- IT resources

  You define the IT resources that MQ Workflow needs to run your business processes. You add definitions for the data and programs you want to use for a process and its activities.

Who is involved in modeling?

Modeling workflow involves different tasks and skills. Buildtime allows you to set up your system to distinguish between these tasks. Several users can be responsible for the different tasks, or the same user can perform several tasks:

System administrator
A system administrator exists in a MQ Workflow system as the first

person. This person is responsible for the initial definition of other staff members. As soon as the workflow model is in place, the system administrator is responsible for maintaining workflow models and monitoring running processes.

Users who define staff
The system administrator can authorize users to create and change the definitions of staff members in the database.

Users who model processes
Users can have the authorization to build and verify the process models. These process models define how to run the processes at run time.

Users authorized for IT tasks
Users can have the authorization to design and define programs to use with MQ Workflow.

What are the modeling steps?

The steps in the modeling process are dependent on one another.

If you complete these steps in the order that is shown, you fulfill the prerequisites for each step.

1. Define the organization of staff members, including roles and levels you need for your organization.
2. Define the network properties for the domain and server components.
3. Define the data structures that a process needs, the activities in a process, and in programs.
4. Register the application programs or tools that you need for the activities in a process.
5. Draw a diagram of a process that shows each activity and block with all the connectors that determine the flow of control and data. Specify the properties for the process.
6. Define, in detail, the logic behind the process diagram:
   - For each activity, specify its start conditions and its end conditions, the persons, data structures, and programs that are required to perform the activity.
   - For each control connector in the diagram, optionally specify a transition condition that is important for the control to flow that way.
   - For each data connector in the diagram, specify how the data in the output container of one activity is mapped to the input container of another.

You translate a workflow model into its Runtime process template by using a Runtime import utility as described in “Chapter 5. Making your workflow model an operational process” on page 43. For details on how to verify a workflow model, see “Verifying a workflow model” on page 44.

The MQ Workflow system and Buildtime

Buildtime is part of an MQ Workflow system and offers a graphical editor for creating process models. Buildtime uses its own relational database to store information about process models. Figure 2 shows the system architecture with Buildtime and its database.

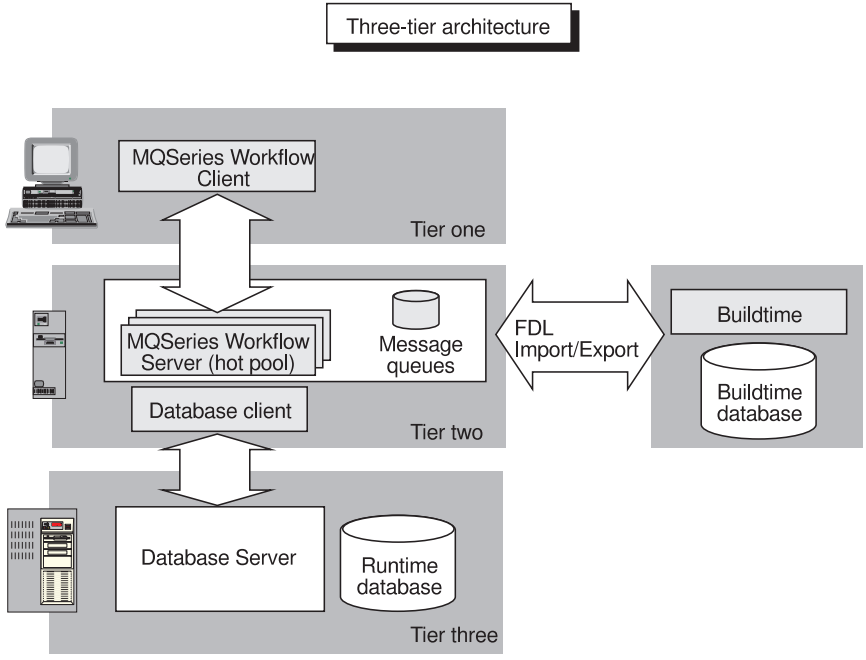


Figure 2. System architecture

For more information about the architecture of MQ Workflow, see *IBM MQSeries Workflow: Concepts and Architecture* .

Instead of using Buildtime to define a new process model you can use model information that is already defined in MQ Workflow Definition Language (FDL). You can import the FDL file into the Buildtime database. For example, this applies if you use a business modeling tool that offers FDL as an exchange format for process models. You can also import these definitions

directly into MQ Workflow Buildtime. For details on how to use FDL, see "Using workflow definitions in Buildtime and Runtime" on page 43.

Using a workflow model in Runtime

After you have created a workflow model in Buildtime and tested it, you export it from Buildtime and import it into Runtime as described in "Chapter 5. Making your workflow model an operational process" on page 43. It is then called a *process template*. In Runtime, for every *instance* of a process, the server components of MQ Workflow navigate through the process. MQ Workflow uses the process model information to move the work to the right person in the right sequence. MQ Workflow starts the programs you defined, keeps process execution history, and provides recovery and restart procedures.

Activities that need to be performed appear on worklists of the MQ Workflow Client of your assigned staff members. When a staff member selects, for example, a program activity, the program starts with the information you defined in your process model.

How Buildtime and Runtime work together

Figure 2 on page 6 shows that Buildtime uses its own database, which is independent from the main database that is used in Runtime. The advantages are:

- The modeler can work independently from the main database, which is the Runtime database.
- You can use the modeling database without any performance impact on running processes.
- You can use a different operating system for your main database, for example, a powerful AIX(R) server. In addition, you might want to use Windows NT or Windows 9x for Buildtime. You might even want to use a mobile system for Buildtime.
- The database tables are optimized for the purpose they serve. The Buildtime database must support long running transactions, whereas the Runtime database can be optimized for the transactional pattern of workflow execution.

When a process is ready for use in Runtime, you must export the model information from the Buildtime database into an FDL file. The FDL file can then be imported into the Runtime database. For details, see "Chapter 5. Making your workflow model an operational process" on page 43.

The design of MQ Workflow will eventually incorporate application program interfaces (modeling application program interface (API)) with check-in and check-out functions to synchronize the two databases automatically. Buildtime is designed to use these APIs and check out data from the Runtime database prior to changing a workflow model. For this release of MQ Workflow you must use export and import utilities of Buildtime and Runtime. If you adhere to the following guidelines, you can be sure that your databases remain consistent.

Guidelines for keeping databases of Buildtime and Runtime synchronized

Follow these guidelines for your databases to remain consistent:

- Create one Buildtime database for your MQ Workflow domain
- Make all definitions you need for your workflow model by using Buildtime
- Use **Mark For Deletion** in Buildtime to prepare the data for deletion in the Runtime database
- Export the data from Buildtime into an FDL file

  Note that Buildtime can selectively export data that is **New**, **In Question**, **Updated**, or **Marked For Deletion** by using the appropriate FDL keywords.
- Use the Runtime Import Utility to import the FDL into the Runtime database
- After you have imported an FDL file into Runtime, update the Buildtime database as follows:
  - Delete the items that you defined as **Marked For Deletion**
  - Unmark the items that you marked as **New** or **Updated** as well as those that have been marked **In Question** after import.

  Buildtime uses symbols to show the *object status*. This indicates the state of an item in your workflow model in Runtime. For details, see "Defining the status of an object for Runtime" on page 43.
- To keep the databases synchronized, do *not* import FDL files into the Runtime database that do not originate from the associated Buildtime database

Because the Buildtime and Runtime databases are not synchronized automatically, you must keep in mind that changes from the Client or via API programming can also lead to inconsistencies between the two databases. These changes can be:

- Changing the password for a user
- Setting the absent flag for a person or resetting it. Resetting the absent flag occurs automatically when the user logs on to the Client
- Changing information for substitutes

To import workflow models into the Runtime database, you must use the Runtime Import Utility as described in “Chapter 5. Making your workflow model an operational process” on page 43.

Chapter 2. Working with Buildtime

This chapter describes how to start Buildtime and how to work with the graphical interface. It assumes that your system administrator has set up your Buildtime installation, using the instructions in the *IBM MQSeries Workflow: Installation Guide*.

Starting Buildtime

Before you start Buildtime, check with the administrator who is responsible for your MQ Workflow installation, how the system is set up for you.

To start Buildtime:

1. Click **Start** on the **Taskbar**.
2. Select **Programs**, then select **MQSeries Workflow**.
3. Click **MQSeries Workflow Buildtime**. If your administrator has created a shortcut for you to start Buildtime, click the shortcut.

Note:

- If you log on to the Windows operating system with a user ID that is also defined in your Buildtime database, you log on to Buildtime automatically. Therefore, no logon window appears.
- If you log on to the Windows operating system with a user ID that is *not* defined in your Buildtime database, the logon window appears. You can then specify a valid Buildtime user ID.

For information about how you can log on with another user ID than the one with which you are already logged on, see the online help topic **Logon**.

Getting help

The online help is the primary source of information when using Buildtime. You find information about the windows in the **Contents** section and more in the **Index** or **Find**.

To get help on a specific field:

- Click the question-mark button, then click the field.

To view all help topics:

1. Click the **Start** button
2. Point to **Programs**
3. Click the **Windows Explorer**

   In the directory where MQ Workflow is installed, you find the help file, which is called **fmcbh***enu***.hlp**, where *enu* represents the U.S. English version. See the *IBM MQSeries Workflow: Installation Guide* for other language abbreviations.
4. Double-click the help file to view the online help topics for Buildtime.

Using the Buildtime interface

When you first start Buildtime, you see the Buildtime window as shown in Figure 3 on page 13. However, there is no diagram displayed.

There is a *tree view* on the left of the Buildtime window that shows all the objects that belong to workflow models. The tabs at the top of the tree view provide a quick way to switch between the different trees. The tabs indicate that you can display object trees for *Processes*, *Staff*, *Network*, and *Implementations*.

The right part of the Buildtime window is a *work area* that is used to display views of workflow elements. This can be the diagram view of a process or the properties that you can define for a selected object.

At the bottom of the Buildtime window, there is a *Status bar*. The status bar shows information such as the name of the database you are using and your user ID.

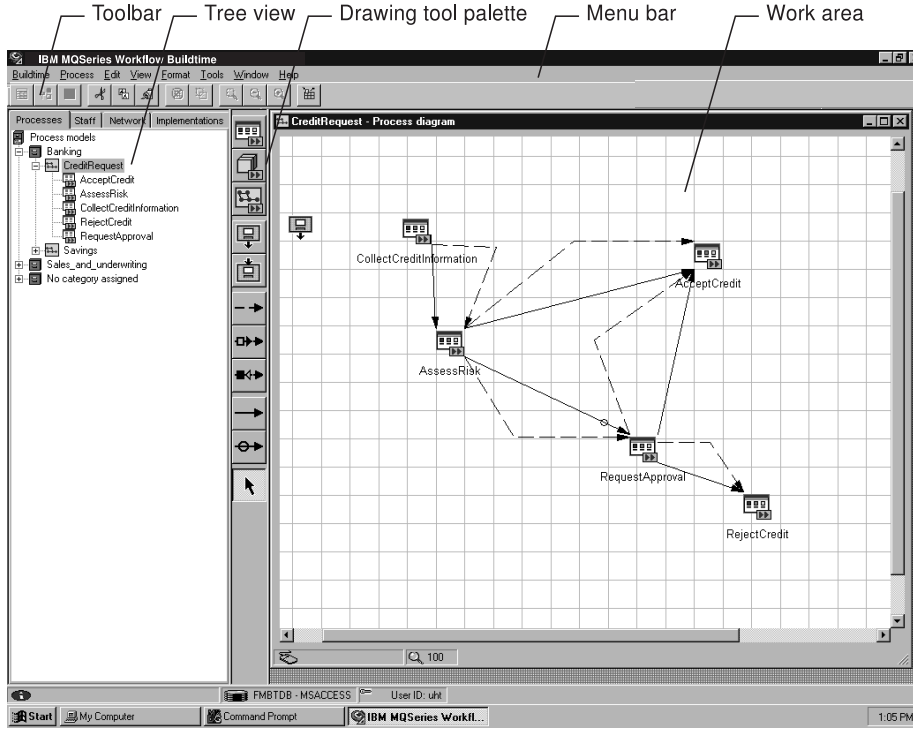


Figure 3. Buildtime user interface

Using views and windows

When you select an object in the tree view, you can choose from different types of views for an object. To choose a view:

- Right-click an element in the tree view, for example, a process that is called *CreditRequest*
- Click **Diagram** in the submenu that opens to show the process diagram for the selected process.

Depending on the element you select in the tree, there can be three different types of views:

Properties

There are property pages that contain the definitions for a selected element. These pages contain tabs to mark the individual pages. For example, you can define and display the properties for a user in the workflow model.

Details

This view is available for elements that are containers for other

elements. For example, you can display the details for processes that belong to a process category in a details view. The details display in a spreadsheet format.

You can even create your own view of Buildtime objects. To use the **Details View Designer**, click **Tools** on the menu bar, then click **Details View Designer**. You can then select an object type and select the properties you want. The Details View Designer starts a query on the objects that are stored in the Buildtime database.

Diagram
The diagram view displays the graphical representation of a process. You can use the diagram editor to create or change a process diagram.

Using the menu bar and toolbar

In addition to the tree view, the Buildtime window contains a *Menu bar* and a *Toolbar*. You can select items from the Menu bar or use the Toolbar as shown in Figure 4. The items in the Menu bar and the Toolbar appear pertaining to the view or window you select. For example, when you click the staff tab, the staff tree opens with items in the Menu bar that you need for specifying staff definitions. Therefore, there is no menu item for Processes.

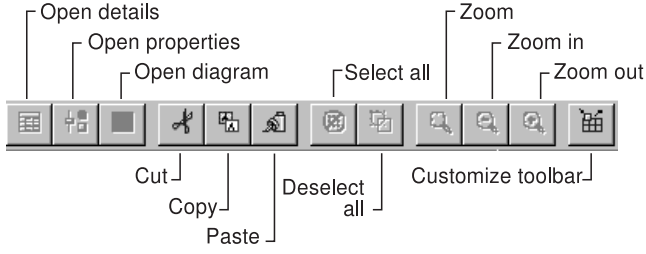


Figure 4. Using the toolbar

For details about the toolbar items, see the online help.

Using and customizing the tool palette

When you open the diagram view of a selected process, the process diagram appears in the right pane. Figure 3 on page 13 displays a diagram view in the right pane. Between the tree view and the diagram, the *Drawing tool palette*, or tool palette for short, appears as shown in Figure 5 on page 15. To display the drawing tool palette, from the **View** menu, click **Drawing Tools**, then select **Show**. If you want to hide the tool palette, click **Hide**. A check mark indicates which feature you selected.

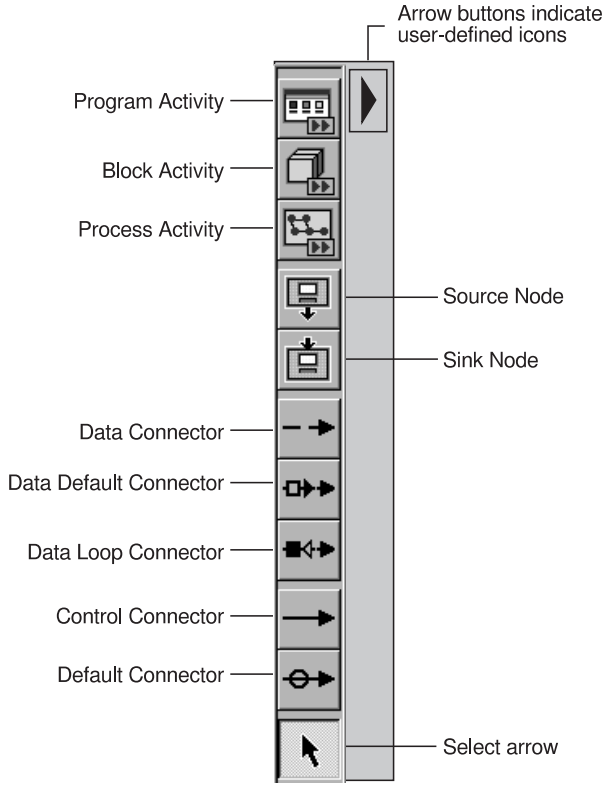


Figure 5. Using the tool palette

If you want to use your own icons for the tool palette, click **Customize**. Depending on how your installation is set up, you can then select your user-defined icons. Arrow buttons indicate that there are user-defined icons available. You can use these icons, instead of the ones offered by default.

For information on how to install your own icons that you can use for drawing your process diagrams, see the *IBM MQSeries Workflow: Installation Guide*. Make sure that you also install your user-defined icons for Runtime using the appropriate installation path.

For information on how to add icons to the tool palette, see the online help.

Using the tree views

You can choose the tree view with which you want to work by selecting the appropriate tab. You can choose to work with the following:

Processes

The tree view for processes shows all the categories, process models, and their activities that are stored in the Buildtime database. The processes are sorted according to the categories that you assign to them. If processes do not have a category, they appear under *No category assigned*.

Staff The tree view for staff shows all the elements that you define for your organization, that is, Persons, Roles, Organizations, and Levels.

Network

The tree view for network definitions shows the system components in a hierarchical order with the domain name at the top of the hierarchy. The system group, system, and servers that belong to an MQ Workflow system structure appear in the tree view.

Implementations

The tree view for implementations shows all the data structures and programs that you define for your workflow.

Chapter 3. Creating a process model

This chapter describes how to define your staff, network properties, data structures, and programs. It also describes how to draw a process diagram.

For information about the process logic that you need to define for the components of your workflow model, see "Chapter 4. Assigning staff and defining process flow" on page 33.

For information about what MQ Workflow verifies when you want to use your process model at run time, see "Verifying a workflow model" on page 44.

Defining your staff

Every process and activity in your workflow model must be associated with one or more persons, identified by their user IDs. That is, every person referred to in a workflow model must already be defined in the Buildtime database. However, you can assign activities to roles. This means that you do not have to assign activities to persons explicitly. Every person who creates a workflow model or an individual process must also be defined. For details on assigning staff dynamically or specifically, see "Chapter 4. Assigning staff and defining process flow" on page 33.

When you select the Staff tab in the tree view, the staff objects are displayed as shown in Figure 6 on page 18. You can work with the properties of the objects that are shown in the tree view as well as add or delete objects.

Planning your staff definitions

If you are the system administrator, you are probably the person who creates the initial staff definitions for your enterprise.

It can be sufficient for your simplest processes that you define only individual staff members. However, to enable processes to support flexible assignment of activities to persons, you can also create the following staff definitions:

- Roles
- Organizations
- Levels

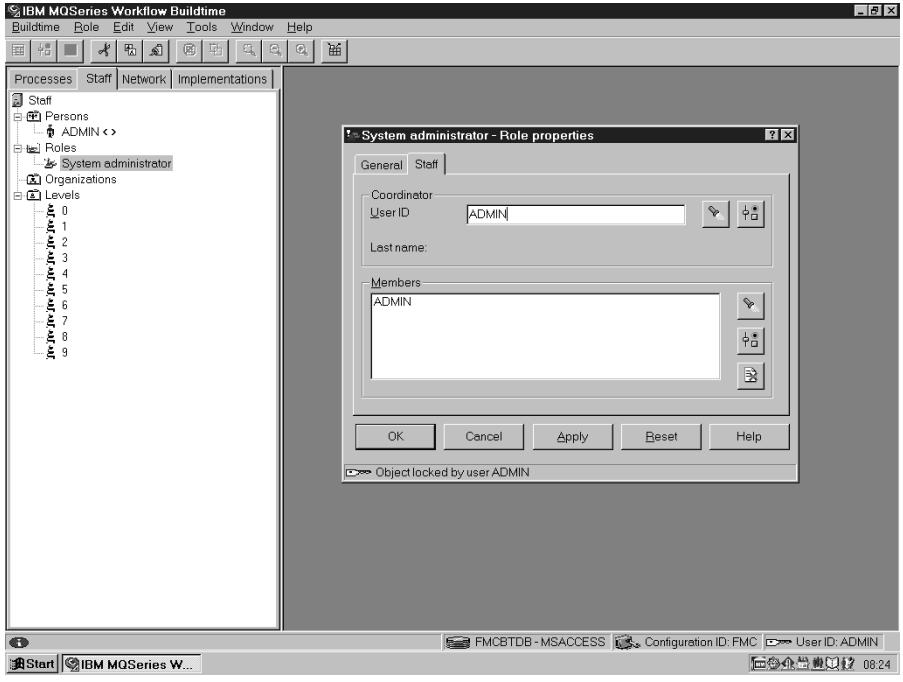


Figure 6. Using the staff tree

By associating one or more of these definitions with the definitions of staff members, you can:

- Establish groups of persons to whom an activity can be assigned
- Assign activities dynamically to persons who meet specific criteria that are related to a level, organization, or role

Define your staff in the following sequence:

- Levels
- Persons
- Roles and Organizations
- Person-role relationships
- Person-organization relationships

Naming levels

You assign levels to persons to distinguish them from each other. You can base these levels on any criteria. You can, for example, assign the highest level to people with the most experience or with the greatest skill.

You can then use level as a criterion when you are filtering candidates for dynamic assignment to activities.

In the **Staff** tree, the **Levels** tree displays 10 levels available to you. You assign a person any level from 0 through 9. These levels are predefined, and you can change only their names and descriptions. You cannot create or delete levels.

To open the properties for a level, from the staff tree view, click the level object with the right mouse button, then click **Properties**. The General page opens, where you can assign a name to the level and enter a description of the level.

Defining persons

To identify to MQ Workflow the persons who are involved in your processes, use the Persons tree. The Persons tree displays the person objects that represent your staff. You can authorize your members of staff for different functions in MQ Workflow. These are then valid in Runtime, when you export them from Buildtime and then import them into Runtime. For details on how to create a person object, see the online help.

Defining roles

The Roles tree displays the role objects used to represent the roles in your enterprise.

A role in MQ Workflow is a function or ability that one person has or that a group of people have in common. For example, this can be a member of a certain working group. One person can have many roles. Many persons can have the same role. When you assign an activity within a process to a role, all persons having this role receive the activity on their worklist in Runtime. Any one of these persons can perform the activity.

When you define roles for the staff in your enterprise, you can also define coordinators for these roles. For example, to define a team in MQ Workflow, you can define a coordinator for a role that is called credit staff. The members of the role that is called credit staff are the members of the team, and the coordinator is the team leader. You can authorize the coordinator to access the worklists of each person who owns the role. Then, at run time, the coordinator can distribute activities among the members of the team credit staff.

MQ Workflow predefines three roles in the Buildtime database, that is, System administrator, Manager, and Coordinator. The system administrator role includes all authorizations for MQ Workflow, and this role must always be assigned at least to one person. However, you can change the assignment from one person to another.

To create a role object, from the staff tree view, click the Roles object with the right button, then click **New Role**. The Role properties are displayed. You can now define the properties for a new role. For details about the input fields, use the online help.

Defining organizations

Organizations in MQ Workflow are administrative units that describe the structure of your enterprise. In MQ Workflow, organizations are arranged hierarchically. An organization can have only one parent organization but any number of child organizations. A person can be a member of only one organization. Note that each organization must have a person assigned as manager.

The Organizations tree displays organization objects to represent these administrative units.

To create an organization object, from the staff tree view, click the Organizations object with the right button, then click **New Organization**. The Organization properties are displayed. You can now define the properties for a new organization. For details about the input fields, use the online help.

Viewing relationships

If you want to view the relationships between persons and roles or between other objects in your Buildtime database, you can open the *Relations viewer*.

To display who is assigned to a specific role, from the **Tools** menu, click **Relations Viewer**. For example, select the object type Role. Select the specific object, for example, Coordinator. To display the user IDs, click **is assigned to**.

Defining your network

For your workflow to be fully operational and run all the activities automatically, you must specify the properties for your MQ Workflow network. The network tree is arranged hierarchically. At the highest level the domain for the MQ Workflow network is displayed.

The workflow model that you define or import into MQ Workflow is valid for the domain. This includes all definitions for staff, data structures, programs, and process templates. You can define properties that specify the behavior of your MQ Workflow installation at this highest level. Whatever definitions you make at the highest level, these definitions are inherited by all lower levels. If you want to have different definitions at a lower level, you can define them explicitly, and the definitions are then valid for that level.

For more information about the architecture of MQ Workflow, see the *IBM MQSeries Workflow: Concepts and Architecture* .

To work with the properties of your network objects, for example, the System Group, do the following:

1. In the Network tree view, right-click the **System Group** object
2. Click **Properties**

   The **System group properties** window appears
3. After you enter the settings, click **OK** to confirm your changes.

For details about the input fields and the syntax of names, use the online help.

Creating a process diagram

You can draw a diagram of a process model with its different types of activities. Typically, there are many individual activities that make up a process. A process can even contain subprocesses and blocks, containing more activities. For details on process and block activities, see "Adding activities to the process diagram" on page 22. In addition to specifying the activities of a process, you also need to specify the control flow and data flow. For program activities you must specify the application programs that you want to use within your process. The staff and network definitions that you need for your workflow to be complete are also part of the process model.

It may be best to get an overview by drawing the whole process model first, including the sequence in which activities must be carried out.

You can specify the properties for the process and its activities as well as the control and data flow after you have created the process diagram. For details, see "Chapter 4. Assigning staff and defining process flow" on page 33 and the online help.

For details about how the steps in modeling your workflow depend on one another, see "What are the modeling steps?" on page 5.

To avoid modeling errors, such as creating endless loops of activities, MQ Workflow uses directed graphs to draw process diagrams. You cannot connect a group of activities to form cyclic flows of control or data. However, you can define an exit condition for an activity that causes the activity to be repeated until the condition is met. You can also define an exit condition for a block that causes a series of activities to be repeated until the condition is met. You can also draw a data connector in a loop for an activity or block that causes data from its output container to be mapped to its input container, while its

exit condition is not met. Activities or blocks that are repeated can therefore have access to data generated during a previous processing of the same activity or block. For details about exit conditions, see “Connectors to control the process flow” on page 37.

Creating a process and specifying its properties

To create a new process, you must define the properties for the process and draw the process diagram. If you want to group your processes according to certain categories, you can define a category. For example, imagine you want to define the process for a banking environment that deals with loan requests; you can define a category for these processes.

To create a new category:

1. In the tree view, right–click **Process models**
2. Click **New Category**
3. Enter a name for the category in the **Category properties** dialog box
4. Click **OK** to save the new category

To create a new process under the category you just created:

1. Right–click the category, for example, *Banking*
2. Click **New Process**
3. Enter a name for this new process
4. Click **OK** to save the new process name

You can now draw the process diagram first, before adding the missing properties for the process. You find details on how to add the properties for a process in “Specifying the properties for a process” on page 27.

Starting to draw a process diagram

To draw a diagram of a new process:

1. In the tree view, right–click the process name you have just created in the tree view
2. Click **Diagram**

The process diagram opens in the work area.

Adding activities to the process diagram

Figure 7 on page 23 shows a diagram view of a newly created process, containing program activities and control connectors indicating the possible sequence of these activities. The process diagram is a representation of your

process that is composed of *nodes* (activities, source and sink containers), and directional connectors (control connectors, data connectors).

The tool palette provides objects to draw all the elements for a process diagram. You can customize the tool palette as described in the online help.

Each object is suited for different tasks and situations, so it is important that you plan your model in detail, before you begin drawing a diagram.

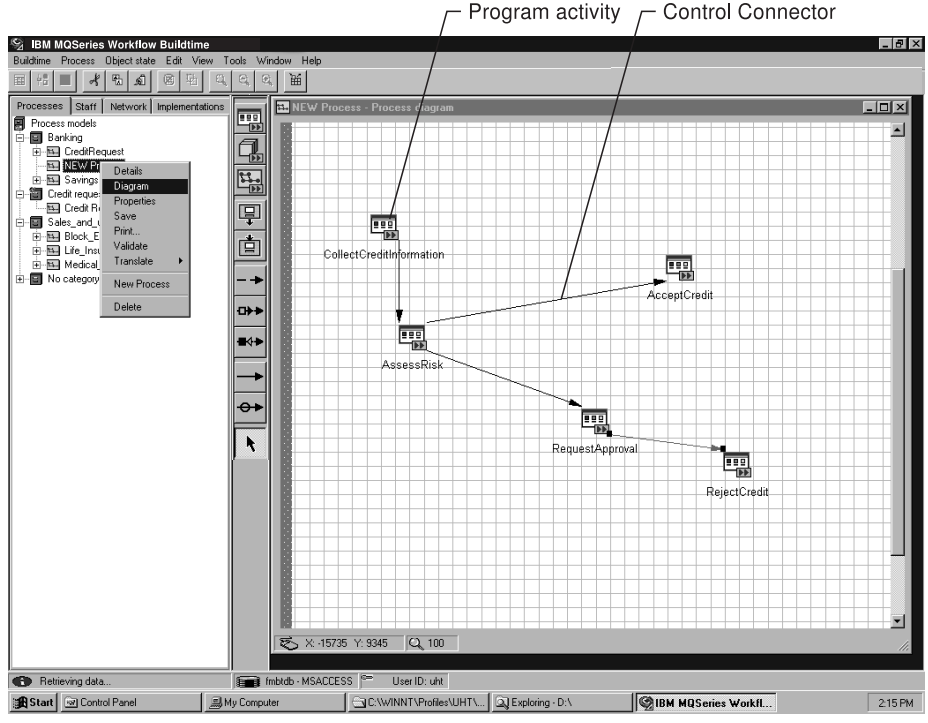


Figure 7. Drawing activities and their control flow

You can use the following icons to model your activities:

Table 1. Icons for activities

	A program activity has a program assigned to it. The program is invoked when the activity is started in Runtime. When the program ends, the program activity's exit condition is evaluated. Depending on the evaluation of the exit condition, the activity either reaches finished status or returns to ready status. If a manual exit is specified for the activity, the person who starts the activity must confirm that it is finished.
	A process activity defines another process (subprocess) that you start from a worklist in Runtime. The process is invoked when the activity is started. When the invoked process ends, the process activity's exit condition is evaluated. Depending on the evaluation, the process activity either ends or starts again.
	A block activity defines a set of activities that can be repeated until an exit condition is met. The block is used to define a do-until loop. You can also define a block to group activities in a complex model.

For details on exit conditions, see "Connectors to control the process flow" on page 37.

To add any of these objects to your process diagram, do the following:

1. Click the symbol in the tool palette that represents the activity you want to add to the diagram.
2. Move the mouse pointer to the position in the diagram where you want to place the node.

   In the drawing area, the mouse pointer changes to the shape of the symbol that you selected on the palette.
3. Click in the drawing area where you want to place the node in the diagram.

You can continue adding nodes of the same type to the diagram simply by moving the mouse pointer to different positions and clicking once. To draw a different type of node or a connector, select its symbol in the tool palette.

If you want to stop adding activities, press the **Esc** key or click the arrow symbol in the tool palette.

Saving a process diagram

To save your diagram in the Buildtime database, do the following:

1. In the **Processes** tree view, select the process you want to save
2. Right-click the process, then click **Save**

For additional information on how to save changes of process properties, see the online help.

Guidelines for drawing a process diagram

As you draw your process diagram, be sure to sequence the activities in your diagram carefully.

1. Draw control connectors to show the order in which activities are to be performed.
2. Draw data connectors to show where output data from an activity is required as input to a later activity. Or draw a data connector as a loop originating from and directed back to the same activity or block if you want output data mapped to input data for repeated executions of the activity or block. You can also map input data to output data.
3. Make sure that each data connector between two activities has a corresponding control connector.
4. Do not draw a control or data connector from a later activity to a previous activity. MQ Workflow prevents you from accidentally creating such cycles in your diagram.
5. If you have a series of activities that should be repeated, put these within a block and specify an exit condition.
6. If your diagram is large or complex, consider using subprocesses to simplify its appearance and reflect orderly levels of complexity. If you want to reuse a set of activities in other processes, you can define these activities in a subprocess. You can use blocks if you have a set of activities that must be repeated until an exit condition is met. The block acts as a do-until loop.
7. If you draw a process activity that starts a process, which contains other process activities, check the sequences of these calls carefully. A process can start instances of other processes in any order, and can start other instances of itself.

To help make your diagram neat, position the activities using a grid on the drawing area. In the **Format** menu, click **Grid** and select **Snap to Grid**.

For information on how to move symbols, see “Moving objects in the process diagram” on page 28.

Joining nodes in a process diagram with connectors

You can add connectors to the diagram between two activities in any combination. The characteristics of connectors are shown in Table 2 on page 26.

Table 2. Connectors in a process

	A control connector specifies the sequence of activities in the process, subject to a transition condition.
	A default connector specifies the sequence of activities if the transition condition of no other control connector leaving the activity evaluates to true.
	A data connector specifies the flow of data from one activity to another.
	A data default connector specifies the flow of data from the input container to the output container of the same activity.
	A data loop connector specifies the flow of data from the output container ″back″ to the input container of the same activity.

For more details, see the online help.

You can add a data connector for these combinations:

- From a source node to an activity
- From an activity to a sink node
- From input to output container of the same activity
- From output to input container of the same activity
- From output container of one activity to input container of the subsequent activity

You must have at least two nodes in your diagram before you can add a connector.

To join activities, do the following:

1. Click the connector symbol you want to use
2. Move the mouse pointer to the activity node, source node or sink node, and click once where you want the connector to start
3. Move the mouse pointer to the target node and click once

   This draws a line between the symbols.

You can bend a connector as you draw it by clicking once where you want to create a bend point. A bend point enables you to continue your connector in a different direction.

You can also delete bend points and you can add them later to existing connectors.

Adding data containers for subprocesses

The source symbol and the sink symbol, as shown in Table 3, represent the data containers that are used to pass input data to and collect output data from a process activity or block activity.

Table 3. Data containers for a process or block

	A source container (input data container) contains data that is to be used as input to a subprocess or block.
	A sink container (output data container) contains data that is to be sent as output from a subprocess or block.

You can have only one input data container and one output data container.

Data containers for program activities are not represented by symbols in a process diagram. For details, see "Chapter 4. Assigning staff and defining process flow" on page 33.

Only process activities and block activities have source and sink nodes.

To add a source or sink node to a diagram, do the following:

1. From the tool palette, click the source or sink symbol to include it in your diagram
2. Move the mouse pointer into the diagram where you want to place the node and click once.

For details on how to move nodes that you have drawn in your diagram, see "Moving objects in the process diagram" on page 28.

Specifying the properties for a process

When you create a new process, the process properties appear. The **General** page opens first, where you specify the name for a process as well as other definitions. For example, you can specify:

Prompt for data at process start
: Select this choice to specify that MQ Workflow should prompt the process starter to initialize data items in the input container of the process that are not set.

When you select the **Data** tab, you can define data structures that describe the input and output data containers of the process. You can drag and drop a

data structure object into these fields to replace the *Default Data Structure* entry. Or you can use the Find button to search for a data structure.

For information on how to specify the logic behind activities in a process diagram, see "Chapter 4. Assigning staff and defining process flow" on page 33. You also find information about data structures in "Defining data structures" on page 29.

For information about what to enter in the fields, see the online help.

Moving objects in the process diagram

In the process diagram, you can move nodes, bend points, and text fields of nodes and connectors.

To move a node, bend point, or text field, do the following:

1. Using the mouse pointer, click the node you want to move
2. Drag the node where you want
3. When you have positioned a node, release the mouse button to drop the node in place

When you move nodes to a different position in the process diagram, the connectors attached to them stretch or contract to adjust to the new position. If you set snapping to grid, each moved node and the bend points of each connector are centered on a grid intersection.

If you change your mind about moving the nodes, you can press the **Esc** key. The objects that you moved return to their original position.

Copying and pasting parts of the process diagram

You can copy and paste segments of a process within one process diagram or from one process diagram to another. If you want to copy connectors, include their origin and target nodes in the segment that you want to copy.

To copy and paste, do the following:

1. Select the segment of your diagram that you want to copy.

   To select a larger segment of the diagram, click the left mouse button and draw a rectangle around the area that you want to copy. Release the left mouse button and all the nodes within the rectangle are selected, including the connectors between them.

   As an alternative, select several nodes and connectors by pressing the Ctrl key as you click the left mouse button on them. You need to press the Ctrl key only while you are clicking the mouse button.
2. Click **Edit** on the Menu bar, then click **Copy**.

The selected part of the process is copied to the Clipboard.

3. Activate the diagram into which you want to insert the process segment.
4. Click **Edit** on the Menu bar, then click **Paste** to insert the copied segment of the process from the Clipboard into the diagram.
5. Move the segment where you want it in the diagram by dragging it.

   If you change your mind about pasting the process segment, you can press the **Esc** key. The process segment is not pasted into the diagram.
6. Click the process segment that you copied to fix it in place.

When you copy a process segment to the Clipboard, it appears there in FDL format (see "Chapter 6. Defining workflow information in an FDL file" on page 63). You can paste this text into a text editor and change the definitions. Then, copy the changed FDL definitions back to the Clipboard, and paste it as a changed process segment into your process diagram.

Notes:

1. When you change an FDL file, choose a text editor that is using the American National Standards Institute (ANSI) codepage to avoid conflicts with differing codepages.
2. When you add an FDL file to the Clipboard, make sure that the file contains an FDL header.

If you want to cut out a segment of a process and move it to a new location within a process diagram, click **Edit** on the Menu bar and then click **Cut**. Follow the instructions for copying and pasting and select **Cut** instead of **Copy**.

Deleting parts of the process diagram

In the process diagram, you can delete nodes, connectors, or bend points.

To delete, do the following:

1. Right-click the object you want to delete
2. Click **Delete** or press the Delete key

To select one item, click on it. To select a group of individual items, click them while pressing the Ctrl key. For more information, see the online help.

Defining data structures

In MQ Workflow, data structure definitions describe the contents of the input and output data containers of processes, activities, and blocks. Any data that is used as input or output, or referred to in exit or transition conditions, must be described in a data structure definition.

Each data structure consists of members. For example, a data structure used to define an address might have members for the street name and the city name.

The data type of a data structure member can be either one of the basic MQ Workflow data types (string, long, floating point, binary) or can refer to another, previously defined data structure. A data structure that refers to another data structure is called a *nested* data structure.

If a data structure A has data structure B as member type and vice versa, however, this cannot work as shown in the following example:

```
STRUCTURE 'A'
    'Member': 'B';
END 'A'
STRUCTURE 'B'
    'Member': 'A';
END 'B'
```

You must define your data structures before you can refer to them in program registrations, process, and activity definitions.

For details about containers and data structures, see the *IBM MQSeries Workflow: Programming Guide*.

Default data structure

MQ Workflow predefines one data structure object, the *Default Data Structure*. When you first click the **Implementations** tab in the tree view, you see this object already created under **Data structures**. You cannot delete or rename the default data structure. You can add user-defined members to the default data structure if you wish.

The properties for every program, process, and block activity contains the default data structure as the default setting for input and output data structures.

You can change the default settings to refer to any other data structure that you create. When the default data structure is changed, the process models to which the data structure is assigned are also changed.

Defining a data structure

To define a new data structure, follow these steps:

1. Create the data structure
2. Specify the properties for the data structure

3. Define the members of the data structure

To create a data structure:

1. In the **Implementations** tree view, right-click **Data structures**
2. Click **New Data Structure**

   The **Data structure properties** window appears.
3. Enter the settings that are including a name for the data structure

For information about what to enter in the fields, see the online help.

Registering programs

The program activities in your model must be able to access these programs at run time. To do this, you must register the programs in MQ Workflow. You must specify the name of the executable program you want to use. In addition, you can specify the following information:

- The environment in which the program must run
- The MQ Workflow input data structure that is used by the program
- The MQ Workflow output data structure that is used by the program
- Any parameters that are to be passed to the program at run time

Because the program activities in your process refer to program registration names rather than real names of programs, your model is flexible. In the registration, you can change the program, the environment characteristics, and the parameters that are passed to the program.

If you change the program that is associated with the program registration, you do not have to save processes again that use the program registration. When you import the workflow model in Runtime (see "Chapter 5. Making your workflow model an operational process" on page 43), the new definitions are valid. However, if you have already imported the workflow model in Runtime, you must export it from Buildtime and import it again in Runtime to use the new definitions.

See the *IBM MQSeries Workflow: Programming Guide* for information about designing applications to run with MQ Workflow and for information about using the application program interface (API).

A program must be defined for each operating system in which it is to be started in Runtime.

To create a program registration object:

1. In the **Implementations** tree view, right-click **Programs**
2. Click **New Program**

The **Program properties** window appears.

3. After you enter the settings, click **OK** to confirm the program registration.

For details about what to enter in the fields, see the online help.

Chapter 4. Assigning staff and defining process flow

This chapter describes how you assign staff to activities and how you can define the process flow. These definitions apply whenever you start a process at run time.

To define the logic behind each activity and connector in a process diagram, you use the appropriate properties window.

To open the properties, for example, a program activity:

1. Open the process diagram of the process for which you want to define the logic
2. Right-click the program activity for which you want to define the properties
3. Click **Properties**

   The **Program activity properties** window appears.

Specifying the properties for an activity

For each activity that you add to your process diagram you must specify properties. These properties determine the process flow at run time.

Select the following tabs to specify the appropriate properties:

- General
- Start
- Exit
- Data
- Tools
- Staff 1
- Staff 2
- Notification
- Control
- Documentation

For information about what to enter in the fields, see the online help.

Assigning staff to an activity

When a Runtime user starts an instance of a process, each activity within this process must have one or more staff members assigned to it.

There are two types of staff assignment:

Dynamic

In dynamic staff assignment, MQ Workflow resolves at run time the criteria you specify here for people who are to receive the activity on their worklists. When the activity becomes ready to start, MQ Workflow Client users who meet these criteria receive the activity.

The criteria you specify can be related to people's levels, organizations, roles, or they can be based on a combination of these criteria. They can also be based on container members that are resolved later in Runtime.

You can also assign the activity to people based on information about the starters of previous activities in the process instance.

The advantage of dynamic assignment is the flexibility it allows you in your workflow model. When changes occur in your staff, you do not need to change your model.

Specific

In specific staff assignment, you specify the user IDs of the people who are to receive the activity on their worklists. Only these people receive the activity.

Assigning activities to specific users is not so flexible as dynamic staff assignment. If an assigned person changes job within your enterprise or leaves it altogether, the assignment becomes out of date. Unless you have assigned some other suitable person to the person object, you must change the workflow model.

However, if you are testing a process, or if there are only certain specific people who can perform an activity, specific staff assignment can be useful.

When you use specific staff assignment and the assignment cannot be resolved, MQ Workflow can change the assignment type for this activity to dynamic. MQ Workflow then tries to resolve the assignment. In this case, the properties specified for the process on the Staff page (role, organization) and for the activity on the **Staff 2** page (roles, organization, level) are used.

Specifying dynamic staff assignment

On the **Staff 2** page, you enter the criteria that a person who starts the activity must meet. Figure 8 shows **Staff 2**, where you can enter these criteria.

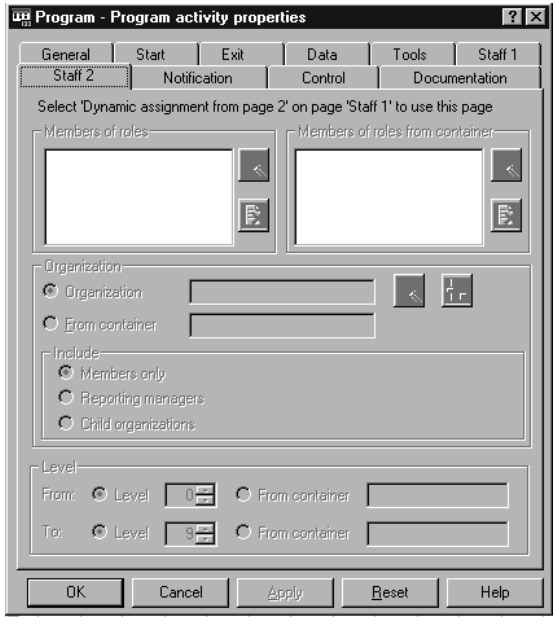


Figure 8. Staff 2 page

On the **Staff 1** page, you must select **Dynamic assignment from page 2** to use the criteria on the **Staff 2** page. Figure 9 on page 36 shows the definitions from which you can choose.

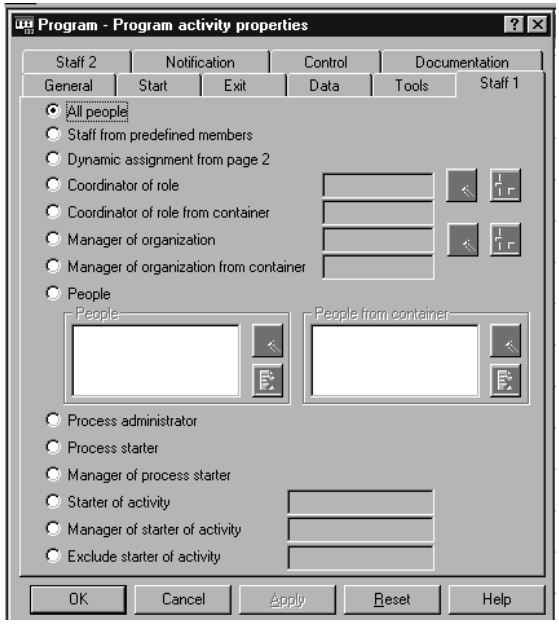


Figure 9. Staff 1 page

If you want dynamic staff assignment based only on the process definitions, select dynamic assignment, but do not select any criteria on the **Staff 2** page. You find a description of the properties for a process in "Specifying the properties for a process" on page 27 and in the online help.

When an instance of a process is started and the activity is ready to start, MQ Workflow uses the criteria for dynamic assignment to identify the group of possible starters for the activity. All the people who meet the criteria receive the activity on their worklists.

If no one meets the criteria or all of those who meet the criteria are absent, the process administrator receives the activity. If authorized to access the activities of other people, the process administrator can transfer the activity.

If you do not specify any criteria, that is, you have activated the selection **Dynamic assignment from page 2** but selected no criteria, at run time the following happens. The activity appears on the worklists of users whose role and organization match those that are specified in the process definition. If no criteria have been specified in the process definition, the activity appears on the worklist of the person who starts the process. The activity also appears on the worklists of everyone else who belongs to the same organization as the process starter.

If you select **Exclude starter of activity**, the staff assignment for the activity must have at least 2 persons defined.

If only one person is defined for an activity *A1*, then this person can only start the activity. As a result of excluding the starter of an activity *A1*, no one is left for the staff resolution for A2.

The following example shows what you must consider when you use this option:

A simple process with 5 activities (A1 to A5) looks like this:
A1 → *A2* → *A3*
→ *A4* → *A5*

For activity *A2* you can have these definitions:

- You can define: Exclude starter of activity *A1*
- You cannot define: Exclude starter of activity *A4* or *A3*

  You can only refer to an activity that precedes the current activity, according to the defined control path.

  If you want to know what you must consider when you assign your staff to activities so that it works successfully at run time, see "Verifying a workflow model" on page 44.

Defining the logic for connectors

To define the logic behind a connector, open the diagram view and double-click the connector in your process diagram. For a control connector, this opens the properties in which you describe the connector. For a data connector, this opens the properties in which you define the data connector.

Connectors to control the process flow

Control connectors determine the flow between activities.

There is a **General** page to specify the name and the description for the control connector. In addition, you can specify the transition condition for the activity:

Transition
: This defines a logical expression you can use for your workflow. When the condition that you specify evaluates to true at run time, control flows to the target of the control connector. Enter a logical expression that describes the condition by using the syntax rules as described in the online help.

If you leave the transition page empty, the transition condition evaluates to true, and the control flow follows this control connector.

If you use the name of an output container variable in a transition condition and do not specify the activity or block name, by default, the activity from which the control connector originates is assumed. If you do specify the activity or block name in the transition condition and later change the name, you must update it here too. Also, if you specify an activity name, there must be a control path from the referenced activity to the current activity.

Connectors to control the data flow

Data connectors determine the flow of data from an originating activity or block to a target activity or block. If the origin and target data structures are the same data structure, and there is no other data connector to the target activity, MQ Workflow automatically maps this data from the origin data container to the target data container.

The user-defined data members appear under the _STRUCT entry in the details view of the data containers.

If the data structures of the two containers are not the same, or there is another data connector to the target activity, you must map the flow of data. If you want to use predefined data structure members, you must also map these. You can also map data from different sources to a single data item.

For details about data structures, see the *IBM MQSeries Workflow: Programming Guide*.

Mapping data between data containers

To begin mapping from an origin activity or block to a target activity or block, right-click the origin activity in the **Processes** tree view. Do this to select **Container Mapping** as shown in Figure 10 on page 40. The container mapping dialog box appears. Data containers for both origin and target activities appear in one window.

The window is split vertically to separate the window into two panes. In the left pane, the origin container appears, and in the right pane, the target container appears. You can scroll both parts independently from each other. If there is more than one origin activity or more than one target available, the container members are listed one below the other.

Origin
This shows the output data structure that is specified on the **Data** page of the properties for the origin activity or block, plus the

predefined data structure members. If the origin of the data connector is the source node of a process or block, the *input* data structure of the process or block is shown.

You map *from* the input container of the process or block to the target input container.

Target This shows the input data structure that is specified on the **Data** page of the properties for the target, plus the predefined data structure members. If the target of the data connector is the sink node of a process or block, the *output* data structure of the process or block is shown. You map *to* the output data container of the process or block from the origin output container.

The types of the data container members must be the same. For example, you cannot map a member of type string to a member of type float.

You can map a complex member, that is, a (user-defined) _STRUCT, a nested data structure, or an array to another complex member if their member items match exactly. A nested data structure and _STRUCT can only be mapped if the target has the same name as the origin. To map other complex data structures with different names directly onto each other, map them member by member.

Alternatively, you can work directly in your process diagram:

- In the process diagram, right-click the activity
- Click **Container Mapping** in the shortcut menu

Figure 10 on page 40 shows the diagram view with data connectors that are drawn between activities and the **Container Mapping** menu in the tree view.

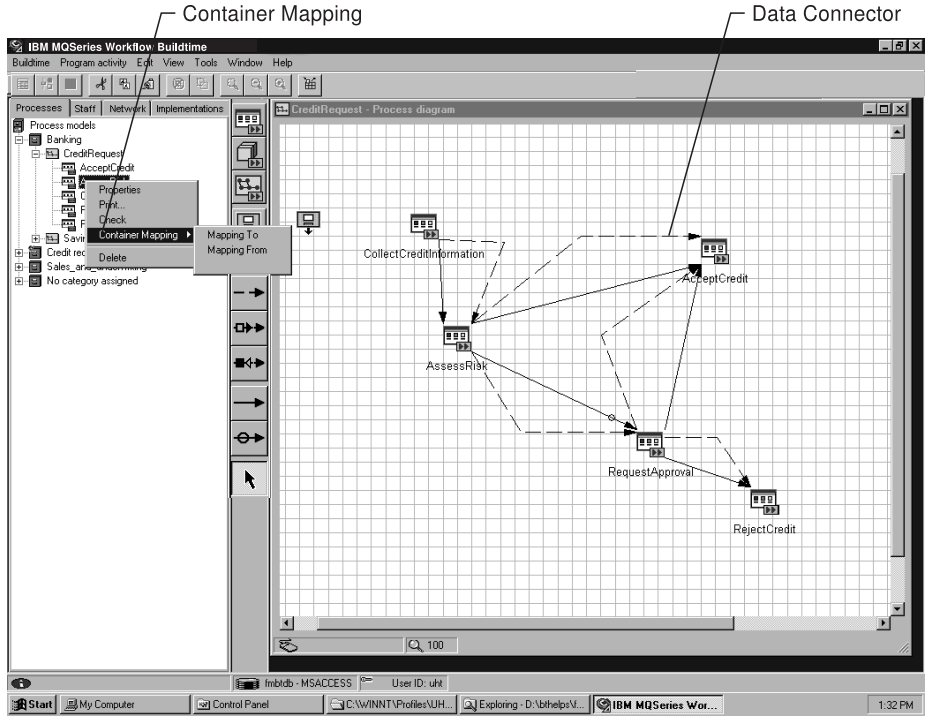


Figure 10. Defining the data flow

Mapping To

In the left pane of the window, the output container of the selected activity appears, for example, AssessRisk. In the right pane, the input containers of the target activities, for example, AcceptCredit and RequestApproval appear.

Mapping From

In the right pane of the window, the input container of the selected activity appears and in the left pane, the output containers of all source activities appear.

If you want to map "inside" an activity, you can use a data loop connector or a data default connector as described in "Joining nodes in a process diagram with connectors" on page 25.

Mapping by drag and drop: To map a data structure member by drag and drop:

1. Drag the data structure member from the Details view of the origin data container

2. Drop it on a data structure member in the Details view of the target data container

You can drop a complex member, that is, a _STRUCT, a nested data structure, or an array on another complex member if their members match exactly. A nested data structure and _STRUCT can only be mapped if the target has the same name as the origin. _STRUCT represents the entire user-defined data structure, but does not include the predefined data structure members.

Mapping predefined data structure members

If you are using the predefined data structure members _PROCESS_INFO and _ACTIVITY_INFO as described in the *IBM MQSeries Workflow: Programming Guide*, you must map these members explicitly from the members in the origin data container to the members in the target data container.

You can also map the fixed predefined data structure members _RC, _PROCESS, and _ACTIVITY, but these appear only in the origin output container. To map them, you must have defined data structure members in the target data structure to which you can map them.

Specifying default values for data container members

For data container members whose type is other than array or nested data structure, you can specify default values to initialize the member items.

To specify a default value for a data container member in the Mapping dialog, you can directly edit the field.

For more details, see the online help.

Chapter 5. Making your workflow model an operational process

This chapter describes how to verify a process model for use at Runtime. It also describes how to export your completed workflow model from Buildtime and import it into MQ Workflow Runtime. These steps are needed to create a process template, from which authorized Runtime users can create executable process instances.

As described in "How Buildtime and Runtime work together" on page 7, there is no automatic transfer of workflow models from Buildtime to Runtime and vice-versa. There is also no automatic transfer for user-defined icons. If you want to use your own icons for your process models, you must install these icons as described in the *IBM MQSeries Workflow: Installation Guide*. Make sure that your user-defined icons are available for both Buildtime and Runtime and that you are using the appropriate installation path for these icons.

Buildtime has built-in functions to export and import workflow model information, whereas Runtime uses a command-line interface, which is part of the Server installation.

This chapter also contains information on how to use a workflow model that was created with MQ Workflow version 3.1 or FlowMark version 2.3.

Using workflow definitions in Buildtime and Runtime

The Runtime database, which is considered to be the main database, is used for running your processes. You define your processes in the Buildtime database or create an FDL file outside of MQ Workflow. When you import FDL files into Buildtime, you can specify if an FDL file originates from MQ Workflow Runtime or from outside of MQ Workflow.

Defining the status of an object for Runtime

To keep the databases synchronized, MQ Workflow uses flags to indicate an object status. For details on how to keep the databases synchronized, see "How Buildtime and Runtime work together" on page 7.

Whenever you want to change definitions that you need for Runtime, make these changes in Buildtime. The same applies if you create new definitions. In

the Buildtime tree view, all major objects are flagged with a symbol to indicate their status. The status symbol is attached to the left of the object in the tree view.

For a list of the object status symbols, see the online help.

Observe the following:

- In an FDL file that you export from Runtime, the status of an object shows that its origin is the Runtime database. When you import the FDL file in Buildtime, then these objects exist in both the Runtime and the Buildtime database.
- If an FDL file is created outside of MQ Workflow Runtime and imported into Buildtime, the status of an object shows the following:

  Updated
  If it is an existing object in Buildtime, it must also exist in Runtime and is therefore considered to be an update

  In Question
  Because there is no indication that the object exists in Runtime, it is considered to be new in Buildtime

If you want to change an object status in Buildtime, you can do the following:

To reset all objects to default:

1. Click **Buildtime** on the menu bar
2. Click **Object Status**
3. Select the new status of the objects

To reset an individual process to default:

1. Click **Object Status** on the menu bar
2. Select the status

For more information, see the online help.

Verifying a workflow model

You can check if the workflow model that you defined in Buildtime or in an FDL file is correct before you translate it for Runtime. When you use the translate option in the Runtime Import Utility, the checks are carried out automatically while translating your model. For details on how to use the translate option, see "Options for the export/import utility" on page 56.

While working with your process diagram, you can start verifying your model as follows:

1. Click **Process** on the menu bar

2. Click **Verify**

This starts a number of checks as described in "Rules for verifying a workflow model".

Rules for verifying a workflow model

When you verify or translate a workflow model, different checks are performed:

- For a process
- For a process and its activities
- For all activities
- For program activities and process activities
- For process activities only
- For block activities only
- For program activities only
- For control connectors
- For data connectors
- For data structures

To make sure that your process model can be used successfully at run time, your model must comply with the following rules:

For a process:

- The diagram cannot be empty. It must contain at least one activity.
- If you define **Duration of process From container**, the data structure member must exist and it must be of type LONG. For details, see "Defining data structures" on page 29 and the online help.
- If you use any of the following definitions, the data structure member must exist and it must be of type STRING:
  - **Process Administrator From container**
  - **Organization From container**
  - **Role From container**

  To create or change these definitions, open the **Process properties** window and click the **Staff** tab. For details on how to define properties, see "Creating a process and specifying its properties" on page 22, "Specifying the properties for a process" on page 27, and the online help.

For a process and its activities

When you define data structures in Buildtime, they serve as *templates* for the data containers at run time. If you specify **From container** for an activity, this means that the data stored in the input container is used at run time for an activity or a process.

The following applies to a process, program activities, process activities, and block activities:

- Input and output data structures must exist. For details on how to define data structures, see "Defining data structures" on page 29 and the online help.
- The number of initial values for both input and output containers is limited as follows: The internal representation of the initial values for each container must not exceed 32 KB.
- For the default values of input and output containers:
  - Read-only predefined input or output container members must not have default values, that is _PROCESS, _PROCESS_MODEL, _ACTIVITY, and _RC . Note that _RC is only an output container member.
  - The input or output container member, for which a default value is to be set, must exist in the related data structure. This includes addressing non-array members as arrays or vice-versa.
  - The input or output container member, for which a default member is set, must have a basic type, that is, it cannot be a substructure or an array.
  - The default value for an input or output container member must conform to the syntax rules of the member type. For example, a string *abc* cannot be assigned to a LONG member.

To change definitions, right-click the data connector. In the shortcut menu click **Mapping**. The container mapping dialog opens. For more information, see "Mapping data between data containers" on page 38.

For all activities

Note: To determine the flow between activities in a process, you use control connectors. The connection from an activity to subsequent activities is called control path.

- The exit condition must be a valid Boolean expression according to the defined syntax of conditions as described in "Syntax of Conditions" on page 70. All data structure members that you use for an exit condition must exist in the output data structure and must have the appropriate type for the context in which they are

used. If a data structure member from another activity is used, a control path must exist from that activity to the activity, which is currently checked.

- You cannot specify more than 254 incoming control connectors.
- If you specify an outgoing control connector with an empty transition condition and there is at least one outgoing default connector, you receive a warning. The warning informs you that the outgoing default connector is never used, because the empty transition condition evaluates to TRUE.
- All data structure members used as substitution variables in the activity description must exist.

For program activities and process activities

The following general rules apply to program activities and process activities:

- All programs that you define as **Support tools** must exist.
- Program properties must be defined for at least one of the platforms. The platforms are: Windows NT, Windows 95, OS/2 (R), AIX, or OS/390 (R).
- If an associated program has an OS/390 external service, the following program properties must be set:
  - **Service**
  - **Service type**
  - **Invocation type**
  - **Executable**
  - **Executable type**
- The number of support tools that you can define is limited. The following formula applies: Take the length of the names of the support tools (number of bytes) and add them to the number of support tools you want to use. The sum must not exceed 254 bytes.
- The checks that apply for program activities also apply for **Support tools**.
- If you use any of the following definitions, the data structure member must exist and it must be of type LONG:
  - **Priority From container**
  - **Duration of activity From container**
  - **Duration of making decision From container**
- If you use any of the following definitions, the data structure member must exist and it must be of type STRING:
  - **Person to notify of delay From container**
  - **Manager of organization from container**

- If you use dynamic staff assignment, the following is checked:
  - If you define **From Level**, you must choose a value that is greater than or equal to 0 and less than or equal to 9. Exception: If you define **From level** with **From container**, the data structure member must exist and must be of type LONG.
  - If you define **To Level**, you must choose a value that is greater than or equal to 0 and less than or equal to 9. Exception: If you define **To level** with **From container**, the data structure member must exist and must be of type LONG.
  - If you do not define **From Level From container** and you do not define **To Level From container**, the value for **From Level** must be less than or equal to the value specified for **To Level**.
  - If you use any of the following definitions, the data structure member(s) must exist and must be of type STRING:
    - **Members of roles from container**
    - **Organization From container**
- If the priority is not taken from the input container, it must be a numeric value ranging from 0 to 9.
- If you use any of the following definitions, the data structure member(s) must exist and must be of type STRING:
  - **People from container**
  - **Coordinator of role from container**
- If you use any of the following definitions, the activity for which you make the choice, for example **Starter of activity**, must exist. In addition, there must be a control path from that activity to the activity, which is currently being checked.
  - **Starter of activity**
  - **Manager of starter of activity**
  - **Exclude starter of activity**

    If you make this choice, the staff assignment for the activity must have at least 2 persons defined.

    If only one person is defined for an activity, then this person can only start the activity. As a result of excluding the starter of an activity, no one is left for the staff resolution. For details, see page 36.

To create or change the properties for program or process activities, open the **Program or Process activity properties** window. Select the **Control** tab for defining Priority or the **Notification** tab to specify duration parameters. For details on how to define activities, see “Adding activities to the process diagram” on page 22 and the online help.

For process activities

- A process must be assigned to a process activity, however it need not exist in the local database. The concept of late binding is applied. This means that the existence of a process is checked only at run time.
- A start activity cannot refer to the same process to which this activity belongs. A start activity does not have incoming control connectors. Other activities can call their own process recursively.

For block activities

- The diagram cannot be empty. It must contain at least one activity.
- The total number of block activities in a process cannot be greater than 32766.
- The maximum nesting level of blocks cannot be more than 100.

For program activities

- A program must be assigned to the relevant program activity and this program must exist.
- If you choose **Program requires these data structures** in the **Program properties** for an associated program, the input data structure of the *program activity* must be the same as the input data structure of the *program*. Equally, the output data structure of the program activity must be the same as the output data structure of the associated program.
- Program properties must be defined for at least one of the platforms. The platforms are: Windows NT, Windows 9x, OS/2, AIX, or OS/390.
- If you define **Program execution server From container**, the data structure must exist and must be of type STRING.
- If an associated program uses an executable program or library (DLL) for Windows NT, Windows 9x, OS/2, or AIX, all members that are used as substitution variables in the command-line parameters of the relevant platform must exist in the program activity's input data structure.
- If an associated program uses an executable or library (DLL) for Windows NT, Windows 9x, OS/2, or AIX library (DLL), the entry point for the platform must be set.
- The definitions you choose for **Program Execution** must fit together as follows:

  If you define **Program execution server**, you must choose **Program can run unattended** in the **Program properties** for the associated program.

- If an associated program has an OS/390 external service, the following program properties must be set:
  - **Service**
  - **Service type**
  - **Invocation type**
  - **Executable**
  - **Executable type**

  To create or change these properties, open the **Program activity properties** window. For defining program properties, click the **Implementations** tab in the tree view and open the **Program properties** window. Then click the relevant tab for the type of data you want to change.

For control connectors

- The transition condition must be a valid Boolean expression according to the syntax of conditions as described in "Syntax of Conditions" on page 70. All data structure members that you use must exist in the output data structure and must have the appropriate type for the context in which you use them. If a data structure member from an activity other than the connector's source activity is used, you must have a control path from that activity to the connector's source activity.

For details on how to define control connectors, see "Connectors to control the process flow" on page 37 and the online help.

For data connectors

- In the **Data Mapping** window, the *From* members of all data mappings must exist in the source data structure, that is, the output data structure of the source activity, unless the connector starts at a **source** node. If the connector starts at a **source** node, the input data structure of the parent block activity or process applies.

  Note: You find the *From* members, which are only those members that are actually used, in the **Mapping column** of the **Target Data Structure** pane.
- The *To* members of all data mappings must exist in the target data structure, that is, the input data structure of the target activity, unless the connector ends in a **sink** node. If the connector ends in a **sink** node, the output data structure of the parent block activity or process applies.

  Note: You find the *To* members in the **Member column** of the **Target Data Structure** pane.

- You cannot use one of the predefined read-only members, that is, _PROCESS, _PROCESS_MODEL, _ACTIVITY, _RC for the *To* members. This applies for all data mappings.
- The source activity of a data connector must be connected to the target activity with a control path.
- The *From* and *To* members of each data mapping must be of the same type. In addition, you can map _PROCESS_INFO to _PROCESS_INFO and you can map _ACTIVITY_INFO to _ACTIVITY_INFO. Note that these predefined members do not have a type.

Note: A warning is issued if a data connector has no data mappings defined.

For details on how to define data connectors, see "Connectors to control the data flow" on page 38 and the online help.

For data structures

- Data structures cannot contain loops. If a data structure A has data structure B as member type and vice versa, this cannot work.

For details on how to define data structures, see "Defining data structures" on page 29 and the online help.

Exporting from Buildtime

The Buildtime export utility enables you to export definitions from this database into an ASCII text file. The exported text file is in a format called Workflow Definition Language (FDL). The syntax of FDL is described in "Chapter 6. Defining workflow information in an FDL file" on page 63. To transfer workflow definitions from Buildtime to Runtime, use the Buildtime export utility first and then import FDL into the Runtime database using the Runtime import utility as described in "Using the Runtime export and import utility" on page 53.

User-defined icons are not exported automatically from Buildtime. If you want to use your own user-defined icons both in Buildtime and Runtime, you must install the icons as described in the *IBM MQSeries Workflow: Installation Guide*.

Starting and using Buildtime export

To start and use the Buildtime export utility, do the following:

1. Click **Buildtime** on the menu bar
2. Click **Export**

This opens the **Export to FDL** dialog.

3. Make your choices for export and click **OK** to start exporting

By default, all definitions in the Buildtime database are shown. You can filter the list of definitions by selecting objects. You can choose which export format you need. As an alternative to FDL, you can choose HTML.

For more details, see the online help.

Importing into Buildtime

To import definitions into the Buildtime database from an FDL file, you can use the Buildtime import utility. You can import workflow information into Buildtime if you want to:

- Restore the contents of your Buildtime database
- Synchronize with the contents of the Runtime database
- Import definitions created outside of MQ Workflow

For details about synchronizing your database, see "How Buildtime and Runtime work together" on page 7.

If you are using the Microsoft Jet database engine for Buildtime, naming must be unique when you import an FDL file.

For example, when you are defining names for roles, processes, data structures, and program registration, you must even distinguish between uppercase and lowercase. You cannot define, for example, *program1* and then define another name *PROGRAM1*.

Starting and using Buildtime import

To start and use the Buildtime import utility, do the following:

1. Click **Buildtime** on the menu bar
2. Click **Import**

   This opens the Import dialog.
3. Make your choices for import and click **OK** to start importing

If the FDL file originates from Runtime, click **FDL from Runtime**.

To avoid overriding existing objects during import into Buildtime, do *not* select Overwrite.

For more details, see the online help.

Using the Runtime export and import utility

The Runtime export and import utility enables you to:

- Export the workflow definitions from the Runtime database into an FDL file
- Import an FDL file into the Runtime database

User-defined icons are not part of the export and import utility. If you want to use your own user-defined icons both in Buildtime and Runtime, you must install the icons as described in the *IBM MQSeries Workflow: Installation Guide*.

The export and import utility is a stand-alone utility, which is started from a command prompt on the MQ Workflow Server.

You can use the utility to:

- Create a new database for Runtime with workflow definitions from Buildtime
- Import and translate workflow definitions from Buildtime
- Import an FDL file that you created outside of MQ Workflow
- Export an FDL file from the Runtime database
- Import and verify an FDL file

Starting the Runtime export/import utility

You can start the utility in two different modes:

- Import mode
- Export mode

To start the utility, log on to the MQ Workflow Server and do the following:

1. On Windows NT or AIX, open a command prompt and change to the directory where MQ Workflow is installed.
2. Enter one of the following in the command prompt window:
   - **fmcibie /i=***in.fdl*

     This starts the utility and imports an FDL file with a file name of *in.fdl*
   - **fmcibie /e=***out.fdl*

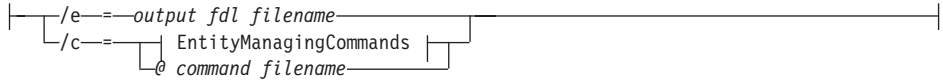
     This starts the utility and exports the definitions into an FDL file with a filename of *out.fdl*

The following syntax diagram shows how to use the utility:

Command syntax of export and import utility

Logon:

Export:



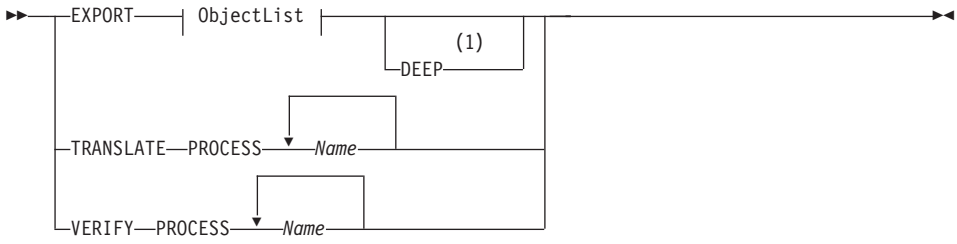
Import:

Log:

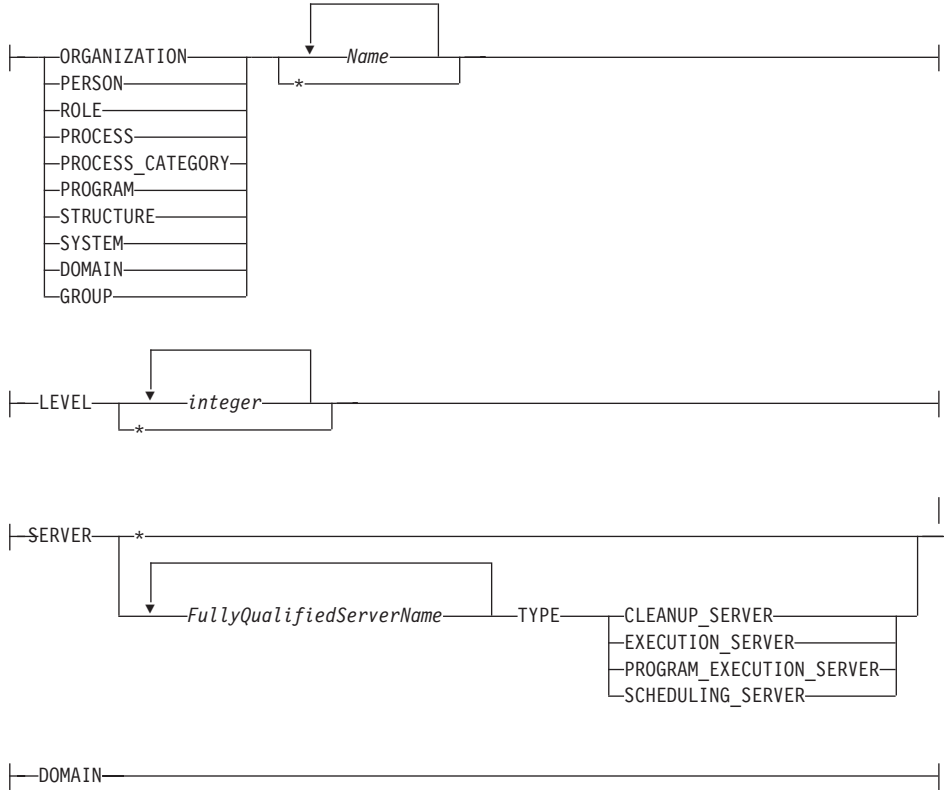
Notes:

1 The specified *-instance* is an MQ Workflow instance.

EntityManagingCommands



ObjectList:



Notes:

1 DEEP is only valid for process.

Note:

- As option delimiter you can use an equals sign (=), a comma (,), a colon (:), or a blank character

- You can specify any one of these options only once
- You can use either option **i** or option **c**, but you cannot specify both
- The extension for the file name of the export log file is **LOG**
- The default destination for the log is stderr (cerr)
- You can use multiple words for option **c** enclosed in quotes

Options for the export/import utility

You can start the export/import utility (fmcibie) using the following options:

Option	Argument	Description
c	Command string	This manages entities commands only in export mode or when starting a file name with an atsign (@). The file you specify contains the commands you want to use.
e	output FDL filename	Exports the specified entities from the database to the specified output FDL file name and is only used for export mode.
h		Displays help information of the utility.
i	import FDL filename	Imports the specified entities from the specified import FDL file into the database.
l	log filename	This specifies the target where the information, warning, and error messages are written to. If you do not specify this option, the default target is stderr. If you do not specify a file name with the option, the log file name is created from the input or output FDL file name by appending the file extension LOG.
o		Overwrites an existing database entity, however, only in import mode.
p	password	This is the password of the specified user ID. The password is optional, when you use unified logon for your installation. Unified logon is only possible on Windows NT.
t		Translates a process model, however, only in import mode. To translate a process model also includes verifying it.
u	userid	This is the logon user ID for the MQ Workflow database.
v		This verifies the process model as described in "Verifying a workflow model" on page 44.
y	instance	The MQ Workflow instance name is used to access the profile settings.

Error codes of the export/import utility

If the export/import utility detects any errors when exporting or importing a file, a return code is specified. If the return code has a value that is greater than 2, the utility stops. As a result of such a severe error, a so-called rollback of the transaction takes place, which means that the database remains unchanged.

Table 4. Error codes of the export/import utility

Value	Description
0	OK - no errors
1	Information message
2	Warning message
4	Validation_Error
8	Syntax_Error - Utility stops
12	Error message
16	Input_Error
20	Severe_Error
24	Internal_Error

Import examples

The following examples show the use of import options:

To import an FDL file

fmcibie /i=*in.fdl* **/u=***admin* **/p=***pwd*

This starts the utility and imports an FDL file with a file name of *in.fdl*, logging on with a user ID of *admin* and a password of *pwd.*

To import and translate a process model

fmcibie /i=*in.fdl* **/u=***admin* **/p=***pwd* **/t**

This starts the utility, imports and translates an FDL file for use in Runtime.

To import and write messages in a log file

fmcibie /i=*in.fdl* **/u=***admin* **/p=***pwd* **/l***log1.log*

This starts the utility, imports an FDL file and writes information in a log file with a file name *log1.log*.

Export examples

The following examples show the use of export options:

To export an FDL file

fmcibie /e=*out.fdl* **/u=***admin* **/p=***pwd*

This starts the utility and exports an FDL file with a file name of *out.fdl*, logging on with a user ID of *admin* and a password of *pwd.*

To export all persons

fmcibie /e=*out.fdl* **/u=***admin* **/p=***pwd* **/c"EXPORT PERSON***"

This starts the utility and exports the definitions for all persons of a workflow model.

To export all persons

fmcibie /e=*out.fdl* **/u=***admin* **/p=***pwd* **/c"EXPORT PERSON** *'ERIC' 'TOM'"*

This starts the utility and exports the definitions for the persons ERIC and TOM.

To export and use commands from a command file

fmcibie /e=*out.fdl* **/u=***admin* **/p=***pwd* **/c@***test1*

This starts the utility, exports an FDL file, and uses the commands from a file, for example, called *test1*. For example, the file *test1* can look like this:

```
Export DOMAIN
Export SERVER *
```

Translate example

To translate an existing model

fmcibie /u=*admin* **/p=***pwd* **/c"TRANSLATE PROCESS** *process1"*

This starts the utility and translates an existing process model in the Runtime database with a process name of *process1*.

Using workflow models of MQ Workflow version 3.1x in version 3.2

If you want to use your workflow model from one of the previous releases of MQ Workflow in this release, you must do the following:

1. Before you install the new release of MQ Workflow, export your Buildtime data as described in "Exporting from Buildtime" on page 51. Make sure that you select **Export all** and **FDL** in the **Export** window.
2. Install the new release of MQ Workflow as described in the *IBM MQSeries Workflow: Installation Guide*.
3. Import the FDL file that originates from step 1. For details on how to import an FDL file, see "Importing into Buildtime" on page 52.

You can now use your workflow model data in the new release of MQ Workflow.

Using workflow models of FlowMark Version 2.3 FDL

If you want to use an FDL file that was created with FlowMark (R) Version 2.3, you must do the following:

1. Import the FlowMark Version 2.3 FDL into Buildtime as described in "Importing into Buildtime" on page 52.
2. Export this updated version of your FDL file as described in "Exporting from Buildtime" on page 51.
3. Import the FDL file into Runtime as described in "Using the Runtime export and import utility" on page 53.

Part 2. Using the external format of MQ Workflow

Chapter 6. Defining workflow information in an FDL file

You define workflow information in a file and then import into MQ Workflow Buildtime as described in "Using workflow definitions in Buildtime and Runtime" on page 43.

This chapter describes the syntax of the declarations and process definitions in the FDL source file.

How to read the syntax diagrams

In this manual diagrams are used to illustrate programming syntax for FDL. To use a diagram, follow a path from left to right , top to bottom, adding elements as you go. In these diagrams, all spaces and other characters are significant.

Each diagram begins with a double right arrowhead and ends with a right and left arrowhead pair.

The following rules apply to the syntax diagrams used in this book:

- The ►►— symbol indicates the beginning of a statement.

  The —► symbol indicates that the statement syntax is continued on the next line.

  The ►— symbol indicates that a statement is continued from the previous line.

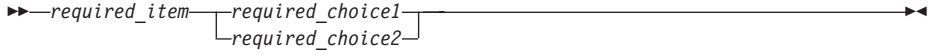
  The —►◄ symbol indicates the end of a statement.

  Diagrams of syntactical units other than complete statements start with the ►— symbol and end with the —► symbol.
- Required items appear on the horizontal line (the main path).

- Optional items normally appear below the main path.

  If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

- If you can choose from two or more items, they appear vertically, in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.



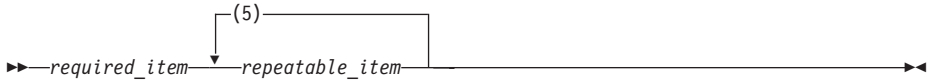
If choosing one of the items is optional, the entire stack appears below the main path.

If one of the items is the default, it appears above the main path and the remaining choices are shown below.

- An arrow returning to the left, above the main line, indicates an item that can be repeated.

If the repeat arrow contains a comma, you must separate repeated items with a comma.

If the repeat arrow contains a number in brackets, the number represents the maximum number of times that item can appear.



A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Keywords appear in uppercase (for example, FROM). Variables appear in all lowercase letters (for example, *column name*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.
- Syntax diagrams may be broken into fragments. A fragment is indicated by vertical bars with the name of the fragment between the bars. The fragment is shown following the main diagram, like so:

A Fragment:

The syntax conventions for FDL

The following sections describe the conventions you must follow when completing the fields in the Buildtime windows or when you create your own FDL files. For information about verifying a process model, see "Rules for verifying a workflow model" on page 45.

The conventions for names and strings are as follows:

Specifying attributes multiple times
If an attribute is specified more than once, the last definition is valid. If exceptions apply, these exceptions are explicitly mentioned.

Quotation marks
Any string that is enclosed in double quotation marks can imbed single quotation marks. A string enclosed in single quotation marks can imbed double quotation marks. Imbedded quotations marks must be duplicated.

Names
Names must be put between quotation marks if they consist of characters other than alphabetic characters (uppercase or lowercase),

numeric characters, or an underscore character. You must put names in quotation marks if they are identical with an FDL keyword.

Size limitations

The following limitations apply to the size of an object. You can specify a maximum length for:

- **Program declaration**: 30720 Bytes
- **Structure declaration**: 30720 Bytes
- **Process declaration**: 4190 KB

Syntax rules for names and strings

To run your process model successfully at run time, observe the rules that apply to the names you can give to MQ Workflow objects. For details about FDL definitions, see "The format of an FDL source file" on page 77. The syntax rules for names and strings are:

ActivityName

The rules for *SymbolName* apply.

Codepage

A codepage number that is specified must be valid and available on the installed system.

Description, Documentation

- You can specify a maximum of 254 characters for *Description* and 4096 for *Documentation*.
- You can specify all characters, except for control characters. You can also specify line ending characters (CR, LF).
- A string that is enclosed in double quotation marks can imbed single quotation marks. A string enclosed in single quotation marks can imbed double quotation marks. However, imbedded quotation marks must be duplicated.

EnvironmentString

- You can specify a maximum of 1024 characters
- You can specify all characters, except for control characters and: & < > ¬ /
- Environment variables have a format of [variable= [string]]

  For more details, refer to the documentation of the operating system.

ExternalContextString

You can specify a maximum of 32 characters. If one of the rules that apply fails, the string is not valid. The rules are:

- Specify a maximum of 32 characters and it must consist of at least one character
- Use only uppercase. Characters that you can use are alphanumeric characters 0 to 9 and A to Z, as well as $ # @
- You cannot use the letters SYS for the first three characters

ExternalShortString

You can specify a maximum of 8 characters. If one of the rules that apply fails, the string is not valid. The rules are:

- Specify a maximum of 8 characters and it must consist of at least one character
- Use only uppercase. Characters that you can use are alphanumeric characters 0 to 9 and A to Z, as well as $ # @ , –
- The first character must be one of the following: A to Z $ # @

ExternalString

- You can specify a maximum of 32 characters and it must consist of at least 1 character
- You can specify all characters, except for control characters and DBCS characters

Float

- You can specify a maximum of 15 characters and it must consist of at least one number
- For floating-point numbers, you must use periods (.) to divide the whole number from the fraction

FullyQualifiedActivityName

- You can specify a maximum of 254 characters
- It must consist of valid *SymbolNames* separated by periods (.)

Level

- You can specify one numeric character. Valid values are: From 0 to 9

Long

- You can specify a maximum of 10 characters and it must consist of at least one number

MappingString

- You can specify a maximum of 254 characters and it must consist of at least 1 character
- You can specify all characters, except for control characters and DBCS characters

MemberName

- You cannot begin a *MemberName* with an underscore character
- The rules for *SymbolName* apply

Name

- You can specify a maximum of 32 characters
- You can specify all characters, except for control characters

ObjectName

- You can specify a maximum of 32 characters and it must consist of at least one character
- You can specify all characters, except for control characters

ObjectShortName

- You can specify a maximum of 8 characters and it must consist of at least one character
- You can use alphanumeric characters 0 to 9 and A to Z (uppercase or lowercase
- You can specify all characters, except for control characters

ParameterString

- You can specify a maximum of 256 characters
- You can specify all characters, except for control characters

PathName and FileName

- You can specify a maximum of 254 characters
- The name must be a valid file name or fully-qualified file name

  For more details, refer to the documentation of the operating system.

PasswordString

You can specify a maximum of 32 characters. The following rules apply:

- The length is less than or equal to 32 characters
- It does not contain control characters
- It does not contain DBCS (double-byte character set) characters

- It does not contain Japanese SBCS–Katakana characters (single-byte character set)

PersonName
- You can specify a maximum of 32 characters.
- It does not contain these characters: @ < > [ ] \ “ ;
- It does not contain control characters
- It does not contain DBCS characters (double-byte character set)
- It does not contain lowercase characters of the local environment
- It is either one of alphabetic (uppercase), numeric, or punctuation character in the current environment or a blank ‘ ’

Priority
- You can specify one numeric character. Valid values are: From 0 to 9

QueueManagerName

The syntax rules apply for:
- OS/2 WARP 4.0
- Windows NT 4.0
- AIX 4.2
- You can specify a maximum of 8 characters.
- You can use alphanumeric characters 0 to 9 and A to Z (uppercase and lowercase), as well as these characters: _ . / % However, / and % are special characters that must be enclosed in double quotation marks.
- You cannot use leading or embedded blanks.
- You cannot use national language characters
- Names may be enclosed in double quotation marks, but this is essential only if special characters are included in the name.

String
- You can specify all characters.
- A string that is enclosed in double quotation marks can imbed single quotation marks. A string enclosed in single quotation marks can imbed double quotation marks. However, imbedded quotation marks must be duplicated.

SymbolName
- You can specify a maximum of 32 characters
- It does not contain these characters: ! ‘ [ ] * + , – . ; / : < = > () \ ∧ “

- It does not contain one of the following keywords, such as: AND, IS, LOWER, MOD, NOT, NULL, OR, SUBSTR, UPPER, VALUE, or the special name _BLOCK and _STRUCT
- It does not contain leading numeric characters
- It does not contain control characters
- It does not contain leading blanks, trailing blanks, or consecutive blanks

SystemQualifier

- You can specify a maximum of 8 characters
- You can use alphanumeric characters 0 to 9 and A to Z (uppercase or lowercase)

WorkingDirectory

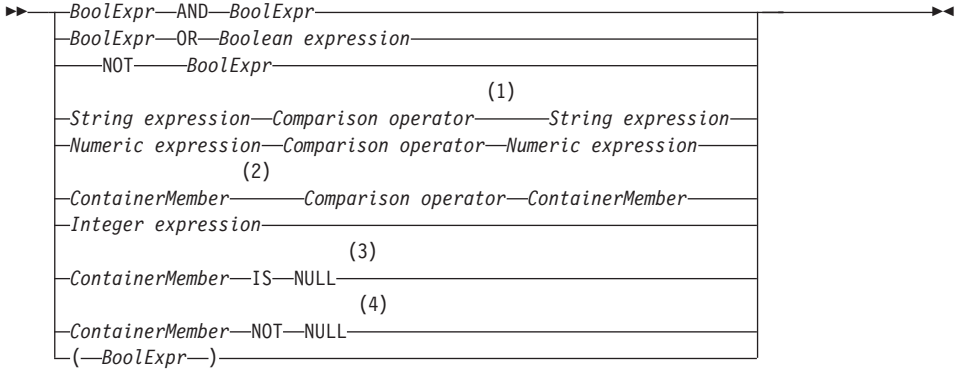
- You can specify a maximum of 254 characters
- The name must be a valid directory name

  For more details, refer to the documentation of the operating system.

Syntax of Conditions

Following are syntax diagrams that describe how to code logical expressions for conditions:

Boolean Expression



Notes:

1 Strings are compared character by character based on the value of their ASCII character codes.

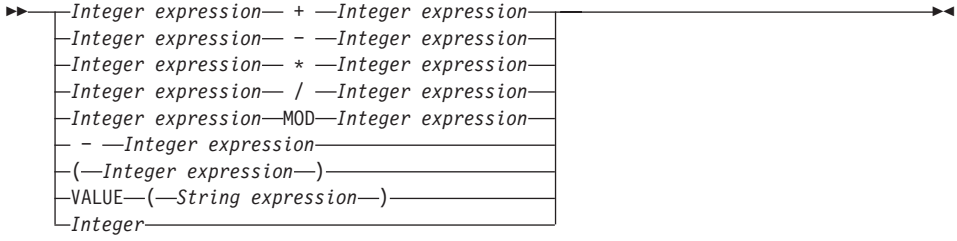
2 Members must be of a basic type, such as Integer, Float, or String. If both container members are NULL, the result is: unknown

3 Use these operators for querying whether a container member is set.

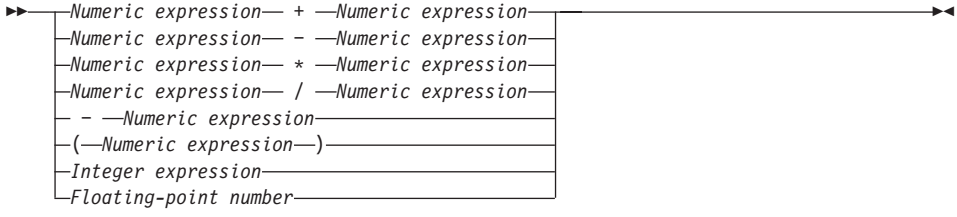
4 Use these operators for querying whether a container member is set.

Comparison operator

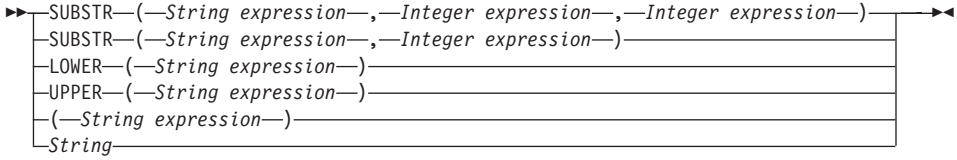
Integer expression



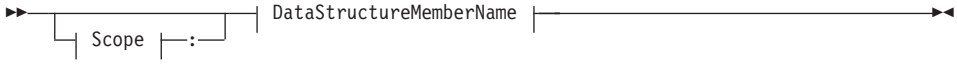
Numeric expression



String expression



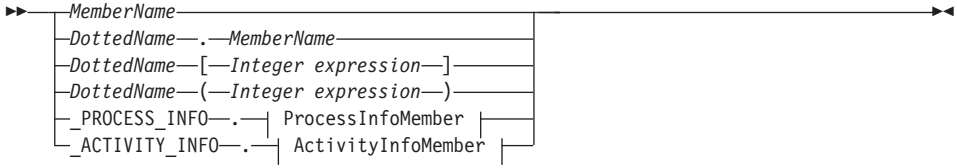
ContainerMember



Scope

DataStructureMemberName

DottedName



ProcessInfoMember

ActivityInfoMember

Notation for exit and transition conditions

Use the following notation in defining exit conditions and transition conditions.

- Functions for use in conditions. The keywords for specifying these functions are *not* case-sensitive. The functions are:

  LOWER
  Function that converts a string from uppercase to lowercase. For example, the following evaluates to true:

  ```
  LOWER("A") = "a"
  ```

  UPPER
  Function that converts a string from lowercase to uppercase. For example, the following evaluates to true:

  ```
  UPPER("a") = "A"
  ```

  SUBSTR
  Function that supports substring extraction, for example,

  ```
  SUBSTR("abcde", 2,3)="bcd"
  ```

  VALUE
  Function that converts a string representation of a number to its numeric equivalent. For example, the following evaluates to true:

  ```
  VALUE("324")=324
  ```

Of course, you can use variables from data containers in place of literal values in any of the preceding examples.

- Special names

  _RC The return code of the activity in its output container. For example, you can use the following for an exit or transition condition:

  ```
  _RC=0
  ```

  This is a long integer.

  _STRUCT
  Refers to an entire default or user-defined data structure. For example, you can represent the contents of the output container of activity A as:

  ```
  A:_STRUCT
  ```

  This is a long integer.

  _BLOCK
  Refers to the current block. For example, you can test the value of the member item

  ```
  ClientFound
  ```

  in the source container of the current block as follows:

  ```
  _BLOCK:ClientFound="No"
  ```

  Of course, you can use variables from data containers in place of literal values in any of the preceding examples.

- Operators. The following list is arranged in order of precedence, from high to low. Operators shown on the same line have the same precedence.

  NOT Unary Boolean "not" operator.

  – Unary arithmetic minus.

  / * Binary arithmetic operators.

  – + Binary arithmetic operators.

  > < = <= >= <>
  Binary Boolean operators.

  AND Binary Boolean operator.

  OR Binary Boolean operator.

  All operators are left-associative, except for the unary minus and the "NOT" operator. Use parentheses, that is (AND), for enclosing parts of expressions to specify order of operations.

- Null operators. These are IS NULL and NOT NULL. Use these operators to query whether a specific data structure member is set.
- Member names
  - To qualify a name in a conditional expression, code the name of the activity to which the condition refers, followed by a colon (:), followed by the member name within the output data structure, for example,

    ```
    UpdateClient:Name.
    ```

    This is optional.
  - To represent a nested data structure member, code the name of the activity followed by a colon, the name of the nested data structure followed by a period, and the name of the data structure member item, for example,

    ```
    UpdateClient:Name.LastName.
    ```
  - For indexing an array, use square brackets ([ ]), for example,

    ```
    Addr.POBOX[0].
    ```
  - For transition conditions, unqualified names in an expression refer to the member name within the output data structure of the source activity of the connector. See "Syntax rules for names and strings" on page 66.
  - For exit conditions, unqualified names refer to the member name within the output data structure of the activity for which the exit condition is defined. See "Syntax rules for names and strings" on page 66.
- You can use predefined data members, which are available in MQ Workflow. To access them, you use the container API. See the *IBM MQSeries Workflow: Programming Guide* for information on using these predefined members. The following types of predefined data members are available:
  - Fixed data members
  - Process information data members
  - Activity information data members
- Numbers. These are floating-point decimals or long integers (32 bit) in decimal, octal, or hexadecimal notation. For integers a leading 0 indicates octal, and a leading 0x or 0X indicates hexadecimal:

  ```
    31     decimal
   037     octal
  0x1f     hexadecimal
  0X1F     hexadecimal
  ```

  Scientific notation may be used for floating-point numbers.
- Strings. These are sequences of any characters in the character set enclosed in single or double quotation marks. If a string literal contains a quotation mark character, this character must be preceded by another quotation mark character.

Evaluation of conditions

The following rules apply for the evaluation of conditions:

- *Short-circuit evaluation*. The evaluation of a condition is stopped as soon as the evaluation of one of its parts determines the result for the whole condition. For example:

  ```
  (FirstName IS NULL) or FirstName='Melissa'
  ```

  If the data structure member FirstName is not set, the expression FirstName IS NULL evaluates to true. Therefore, the complete condition evaluates to true. The second expression is not evaluated.
- *Three-value logic* Besides true or false, a condition can also evaluate to *unknown*. For example:

  ```
  FirstName='Melissa' or (FirstName IS NULL)
  ```

  If the data structure member FirstName ist not set, the expression FirstName=’Melissa’ evaluates to unknown. The second expression evaluates to true. Therefore, the complete condition evaluates to true.

  See Table 5 on page 77 for the truth table for conditions with Boolean operators AND, OR, and NOT.

Table 5. Truth table for AND, OR, and NOT operators

		a AND b	a OR b	NOT a
a=t	b=t	t	t	f
	b=f	f	t	-
	b=?	?	t	-
a=f	b=t	f	t	t
	b=f	f	f	-
	b=?	f	?	-
a=?	b=t	?	t	?
	b=f	f	?	-
	b=?	?	?	-

Legend:

f false

t true

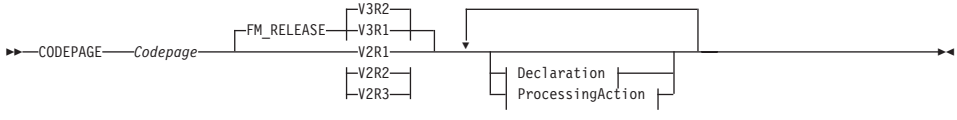
? unknown

The format of an FDL source file

The FDL source file contains process descriptions written in the FDL language. The FDL source file starts with any number of declarations, followed by any number of process definitions.

For details about syntax rules for names and strings that you can use to define objects, see "The syntax conventions for FDL" on page 65.

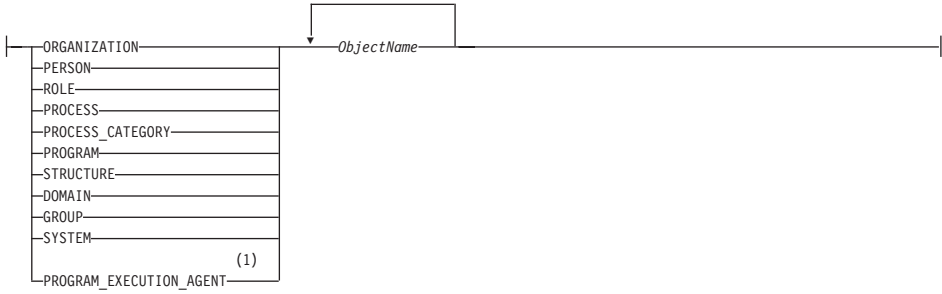
FDL source file



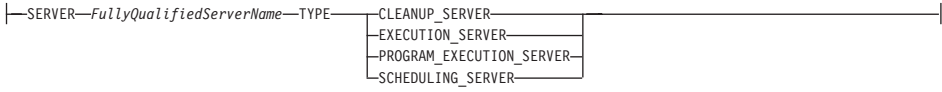
Declaration:

ProcessingAction:

ObjectList:



ObjectServer:



Notes:

1 The name of PROGRAM_EXECUTION_AGENT is the *-PersonName* of the RELATED_PERSON attribute.

Tips for the processing actions CREATE, REPLACE, UPDATE, DELETE:

CREATE is the default processing action. A new entity is created in the database. If the entity already exists, the system issues an error message. Note that the overwrite option -o, as described in "Options for the export/import utility" on page 56, is only valid for the processing action CREATE. When you specify -o and the entity already exists in the database, the processing action is changed automatically to REPLACE.

REPLACE allows you to entirely replace an existing entity in the database. If the specified entity does not exist, the systems issues an error message.

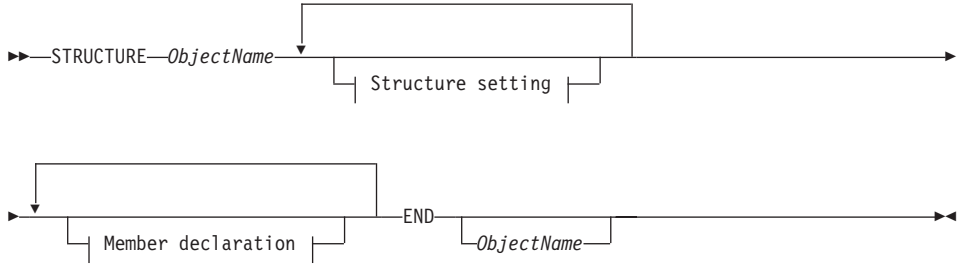
UPDATE requires that the entity must exist in the database. The specified attributes are updated. The value is changed if a single value applies. If you can specify multiple values, which is indicated as a repeatable item, the specified values are added to the existing ones. For example, if you specify

```
UPDATE ROLE R1 RELATED_PERSON P1 END
```

the person P1 is added to the role. The exception to the rule is that the predefined ROLE System Administrator can only have one related person. Therefore, if you update the RELATED_PERSON attributes, you replace it altogether.

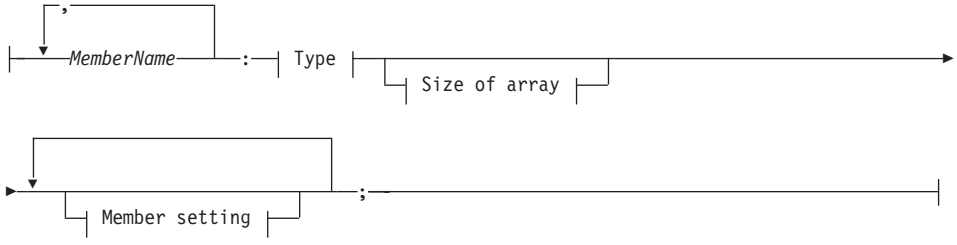
DELETE is used to completely delete an exisiting entity in the database.

Data structure



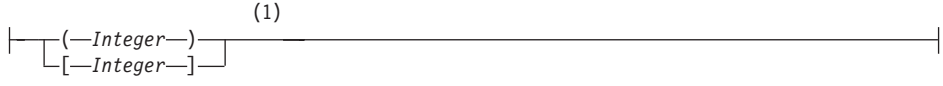
Structure setting:

Member declaration:



Type:

Size of array:

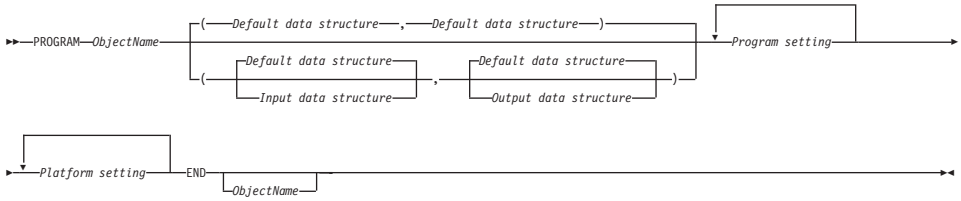


Member setting:

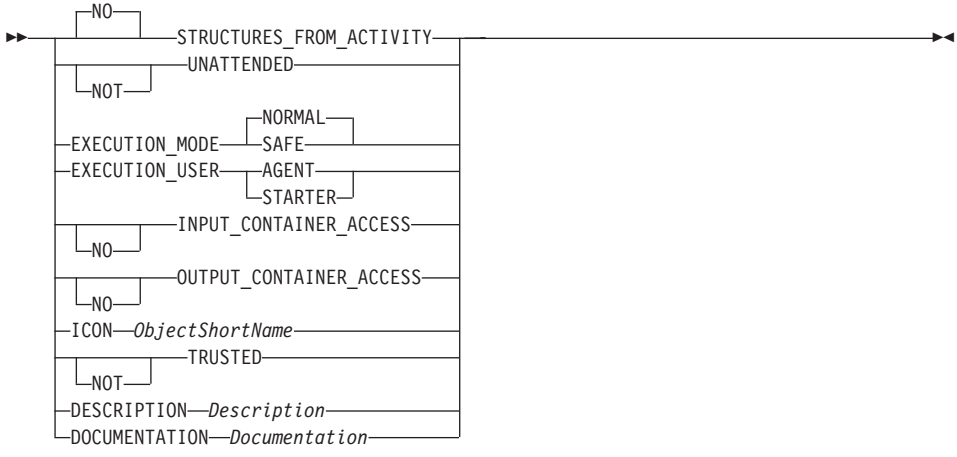
Notes:

1 The maximum size that you can specify is 512 elements.

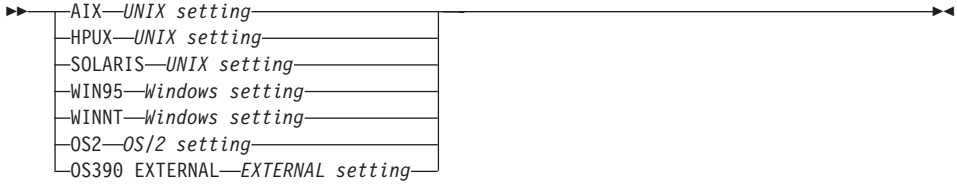
Program



Program setting



Platform setting

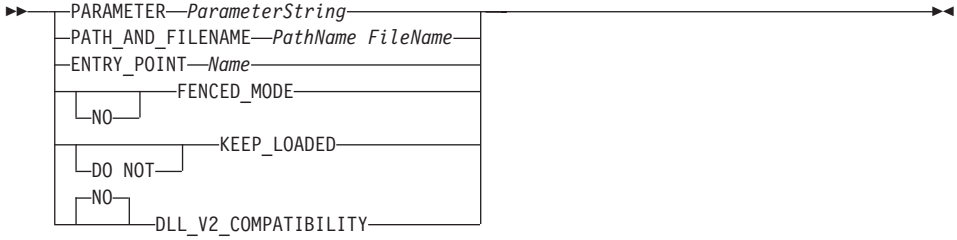


UNIX setting

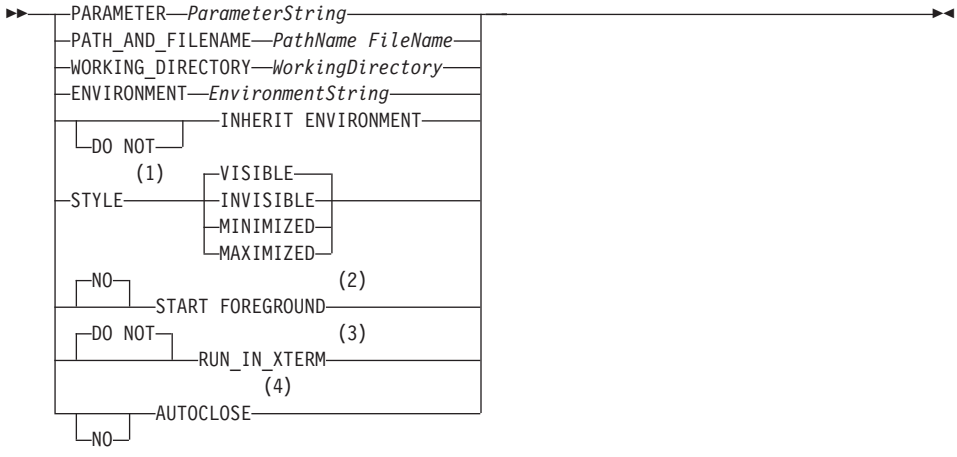
Windows setting

OS/2 setting

DLL setting



EXE setting



Notes:

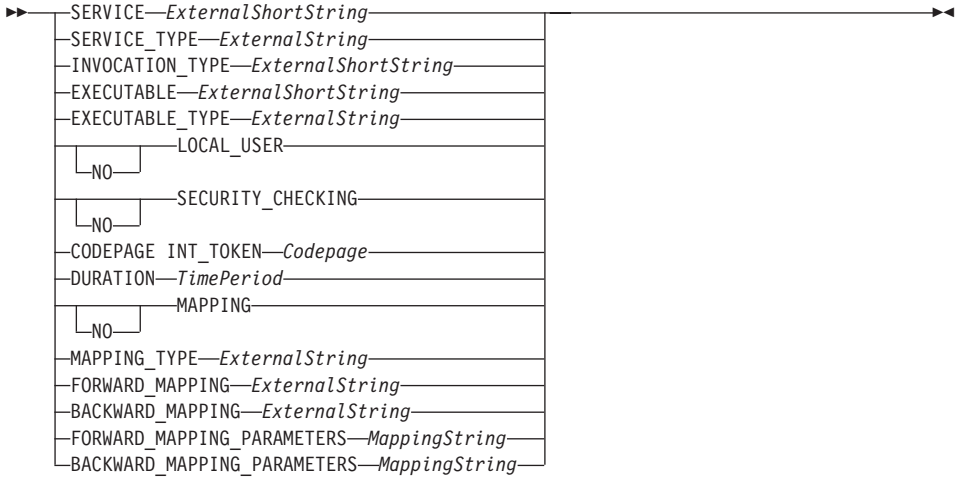
1 Windows 95, Windows NT, and OS/2 only

2 Windows 95, Windows NT, and OS/2 only

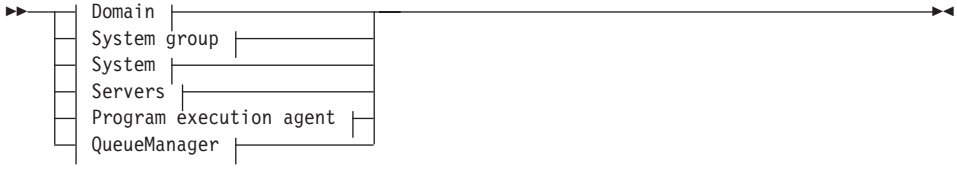
3 UNIX only

4 OS/2 only

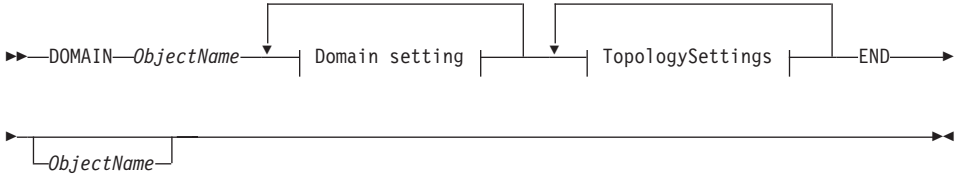
EXTERNAL setting



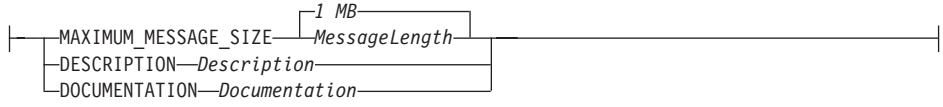
Topology



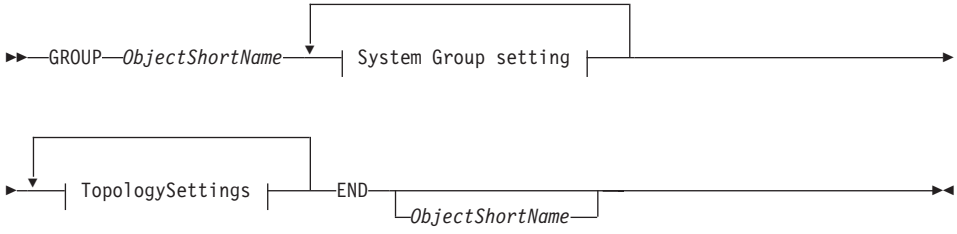
Domain



Domain setting:

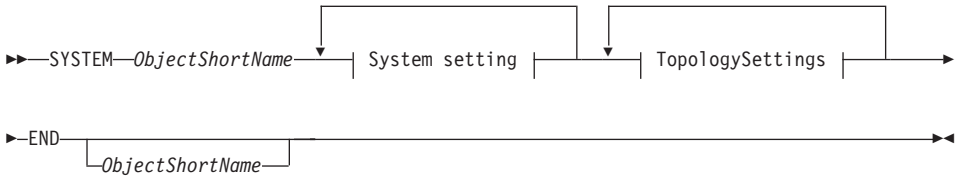


System Group

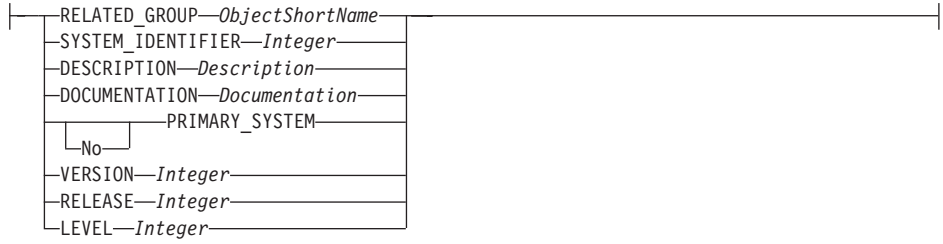


System Group setting:

System



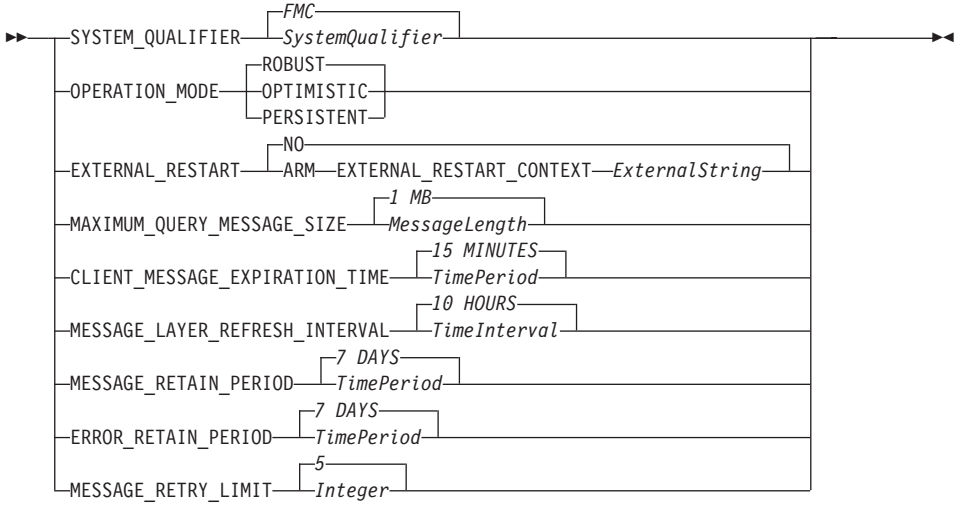
System setting:



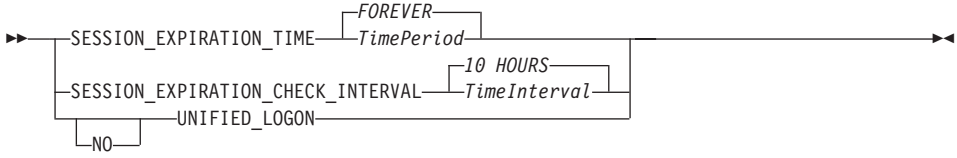
TopologySettings

The default values apply for the domain only, because these attributes are mandatory. For other hierarchical levels, that is, System Group and System, no default values are set, because they are optional.

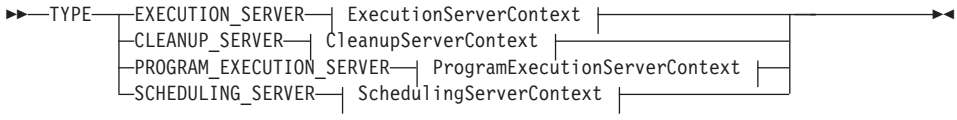
OperationSettings



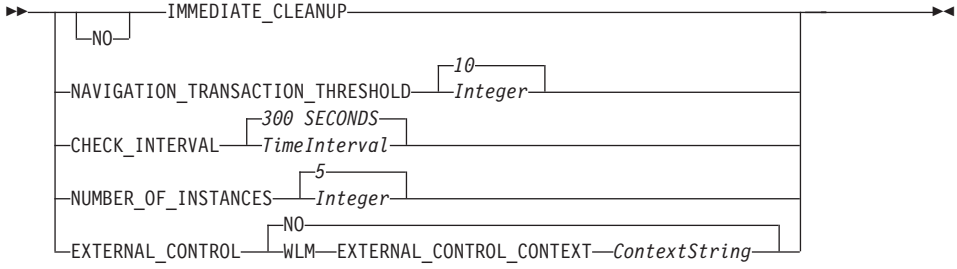
SessionSettings



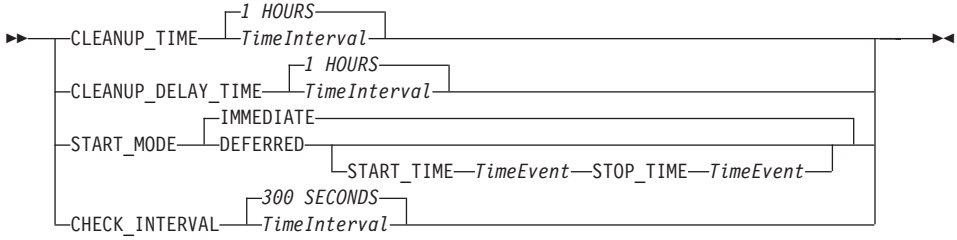
ServerSettings



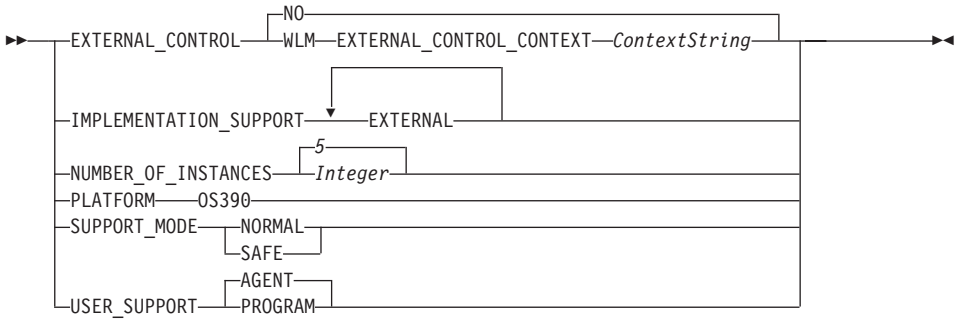
ExecutionServerContext



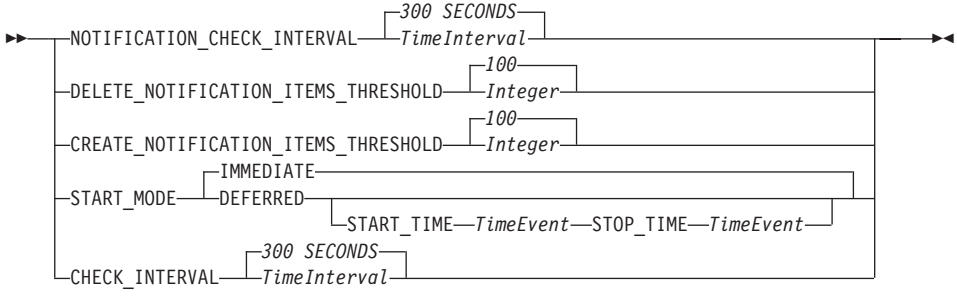
CleanupServerContext



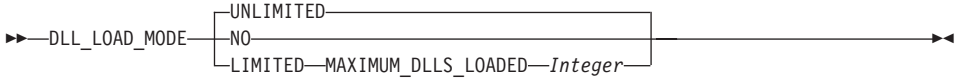
ProgramExecutionServerContext



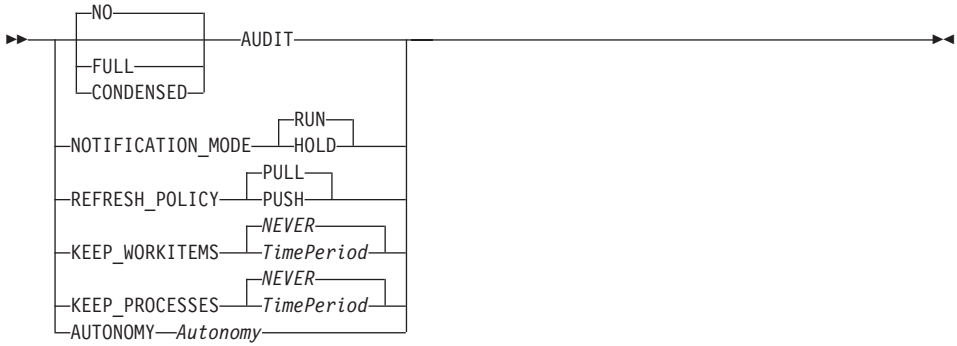
SchedulingServerContext



DefaultProgramExecutionAgentSettings

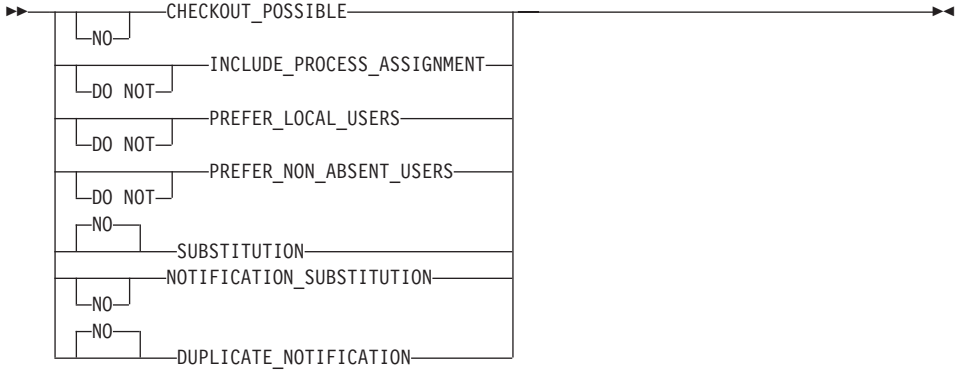


DefaultProcessSettings

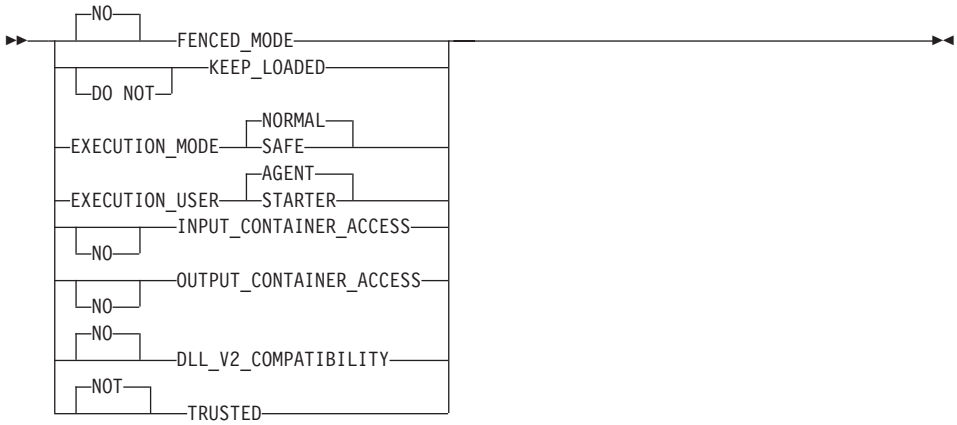


Autonomy

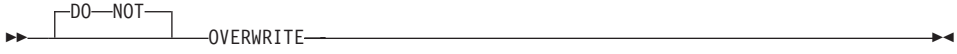
DefaultActivitySettings



DefaultProgramSettings

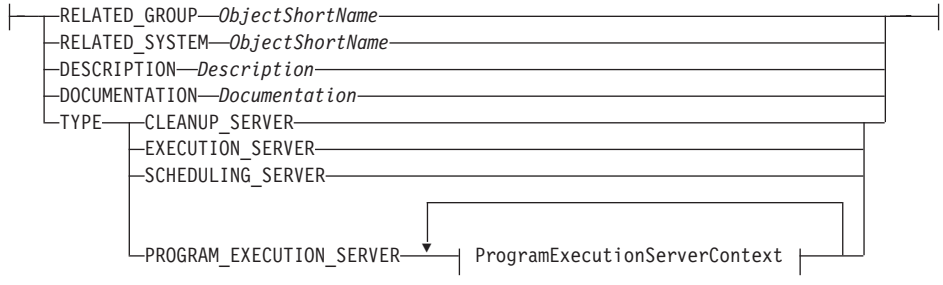


DefaultImportSettings



Server

ServerSetting:



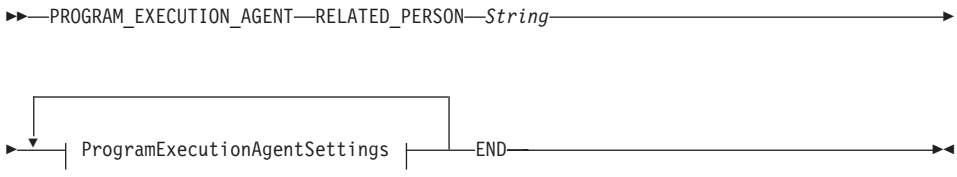
Notes:

1 The *-ObjectShortName* for a server can only be one of the following:

- -EXECSVR for EXECUTION_SERVER
- -CLEANSVR for CLEANUP_SERVER

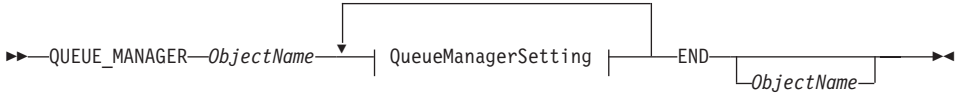
- -PESERVER for PROGRAM_EXECUTION_SERVER
- -SCHEDSVR for SCHEDULING_SERVER

ProgramExecutionAgent

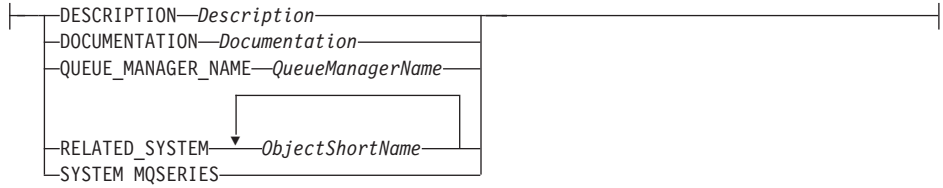


ProgramExecutionAgentSettings:

QueueManager

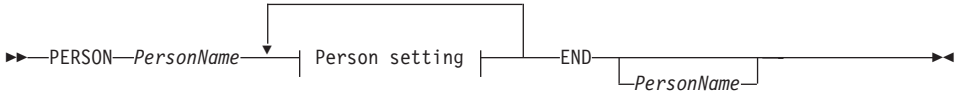


QueueManagerSetting:

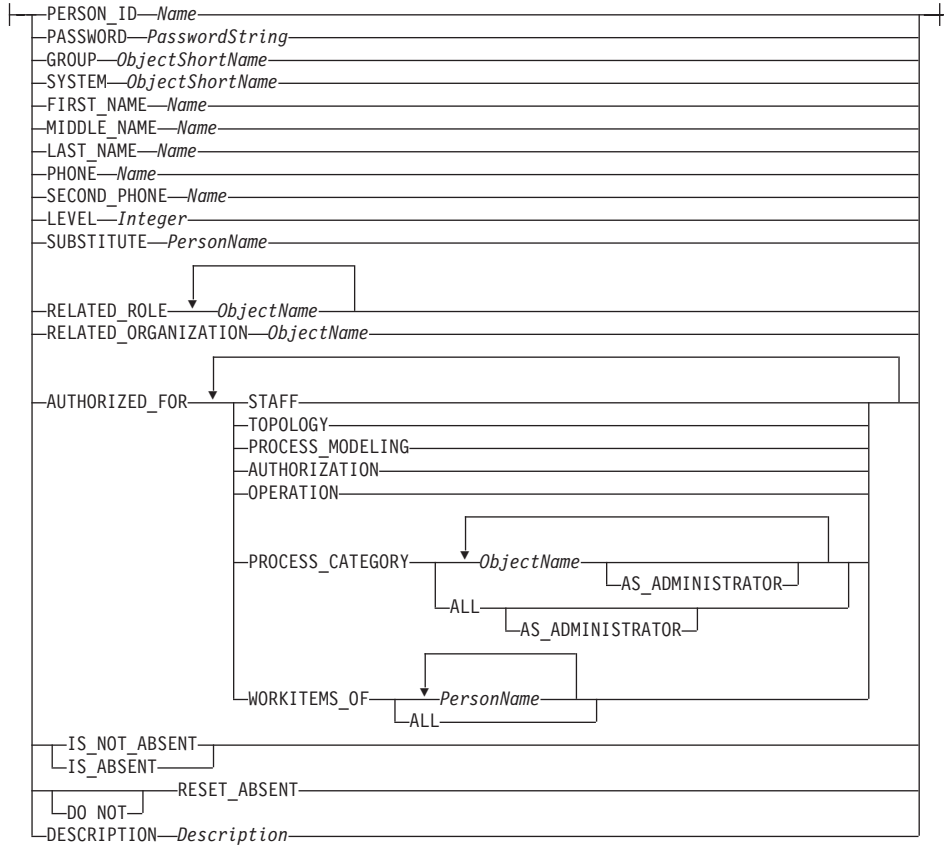


Staff

Person



Person setting:



Tips for the Person setting:

You can specify the **AUTHORIZED_FOR** attribute more than once. All the values that you specify are then valid for that person. However, the following restrictions apply for the PROCESS_CATEGORY as shown in these examples:

- You can specify

  ```
  AUTHORIZED_FOR PROCESS_CATEGORY ALL AS_ADMINISTRATOR
  ```

- You can specify both

  ```
  AUTHORIZED_FOR PROCESS_CATEGORY ALL
  AUTHORIZED_FOR PROCESS_CATEGORY 'cat1'AS_ADMINISTRATOR 'cat2' AS_ADMI
  ```

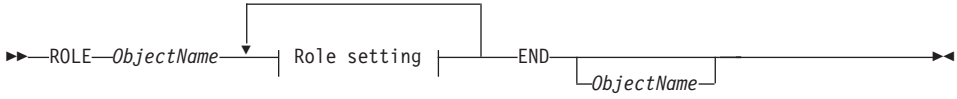
- You can specify

  ```
  AUTHORIZED_FOR PROCESS_CATEGORY 'cat1' AS_ADMINISTRATOR 'cat2'
  ```

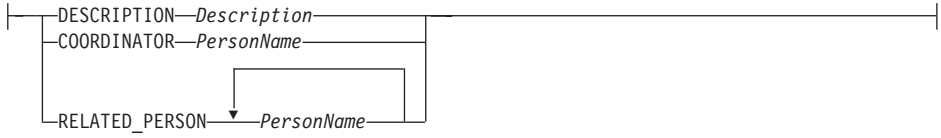
- You can both specify

  ```
  AUTHORIZED_FOR PROCESS_CATEGORY 'cat1' 'cat2'
  AUTHORIZED_FOR PROCESS_CATEGORY 'cat3' AS_ADMINISTRATOR
  ```

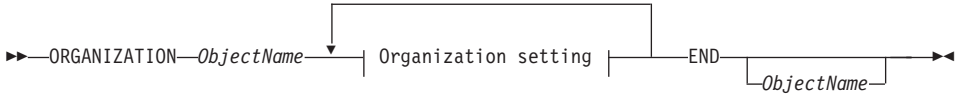
Role



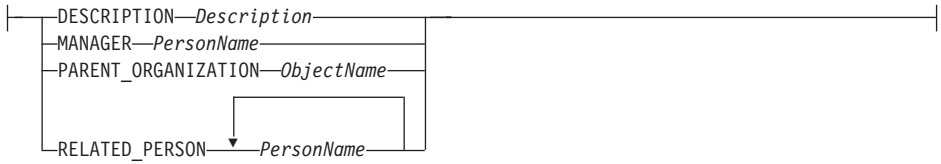
Role setting:



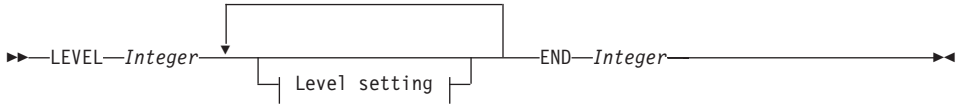
Organization



Organization setting:

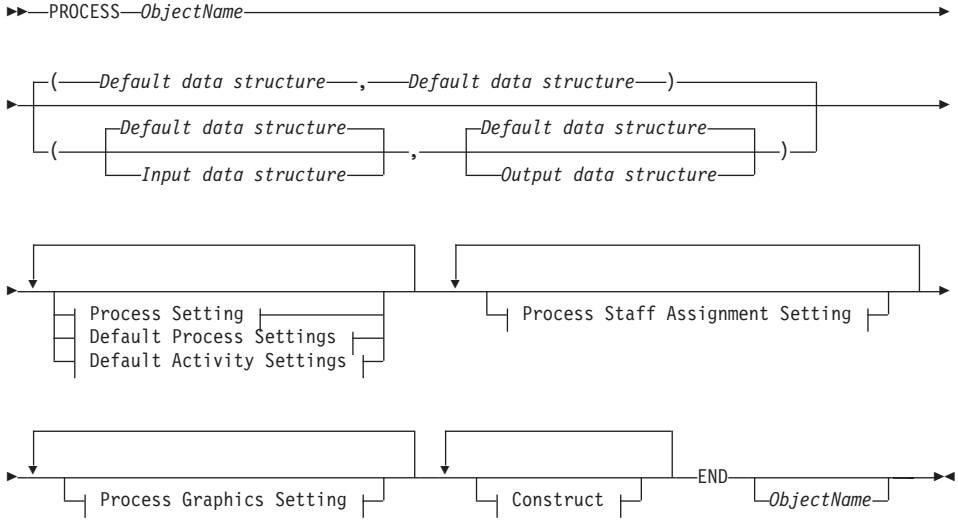


Level

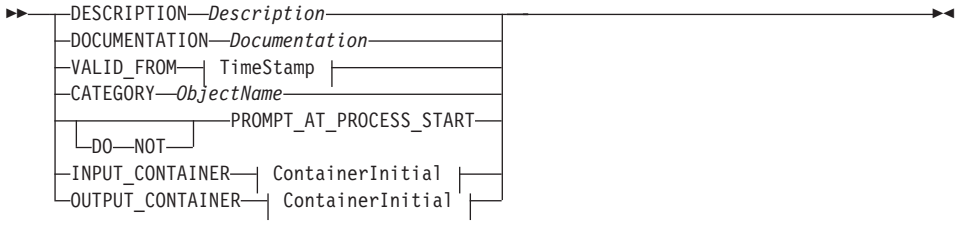


Level setting::

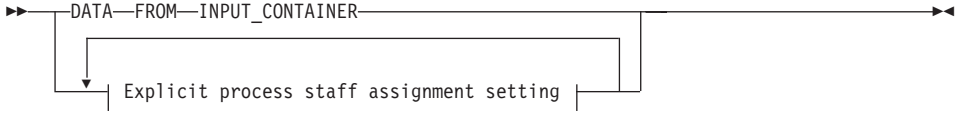
Process



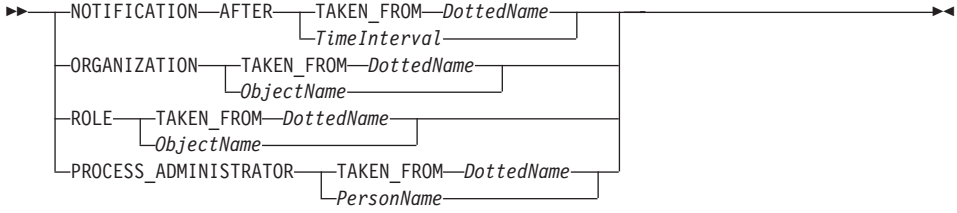
Process Setting



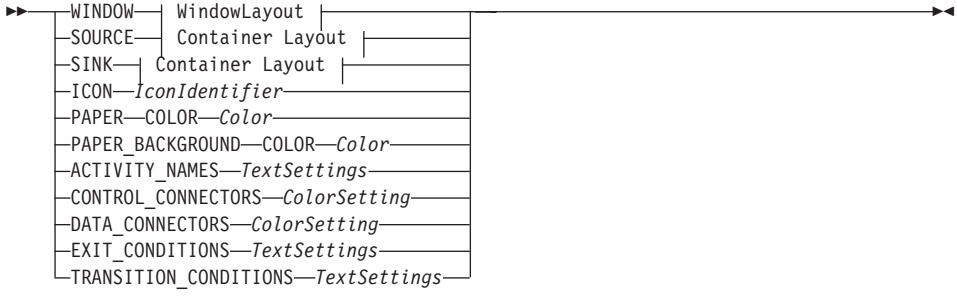
Process Staff Assignment Setting



Explicit process staff assignment setting



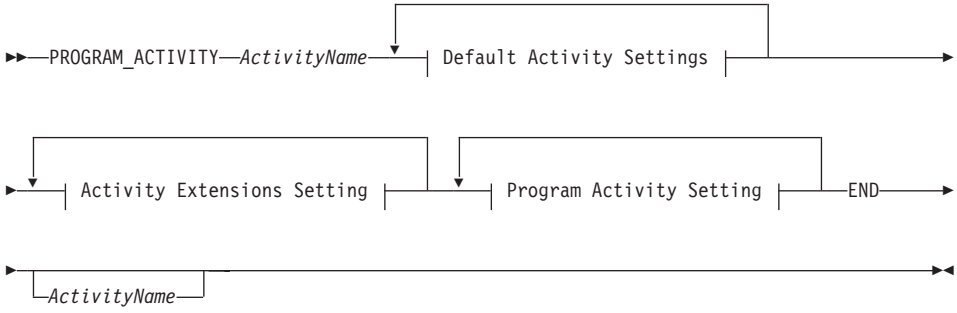
Process Graphics Setting



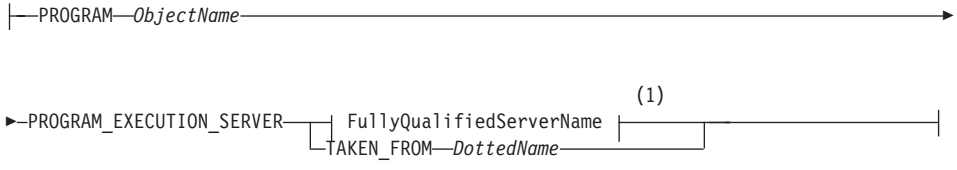
Construct

Activity

Program activity



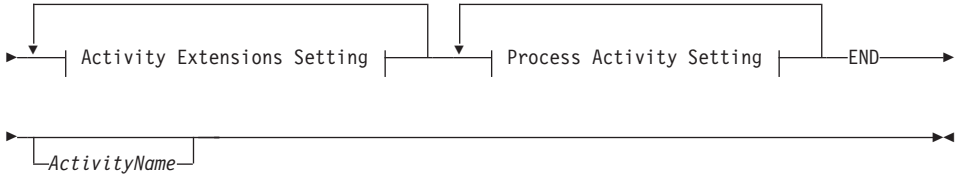
Program Activity Setting:



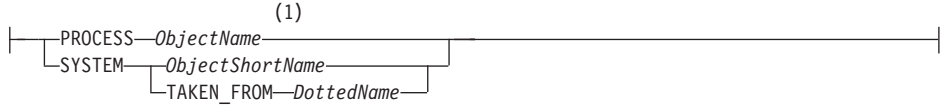
Notes:

1 The type of server can only be an Execution Server.

Process activity



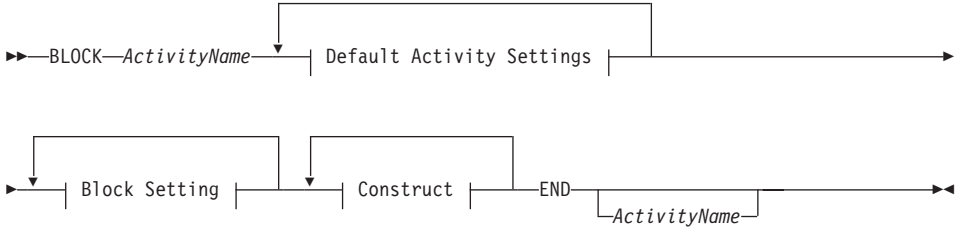
Process Activity Setting:



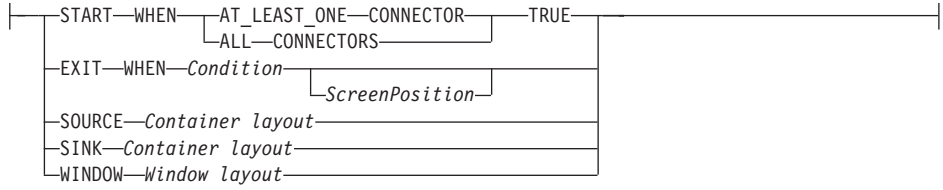
Notes:

1 This is the name of a process

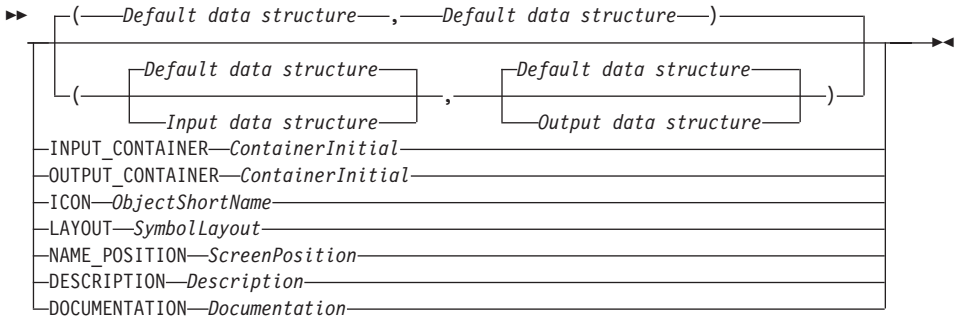
Block



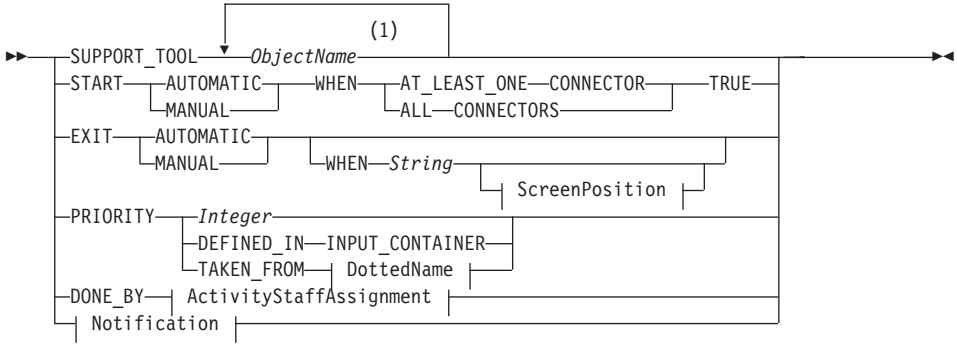
Block Setting:



Activity Setting



Activity Extensions Setting

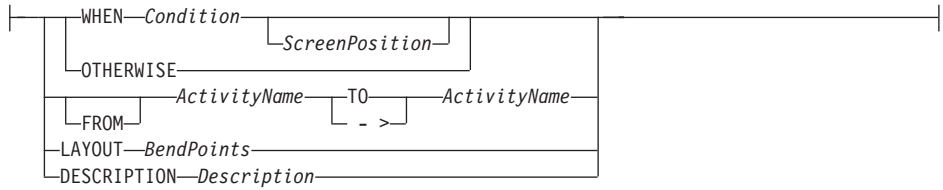


Notes:

1 This is the name of a program.

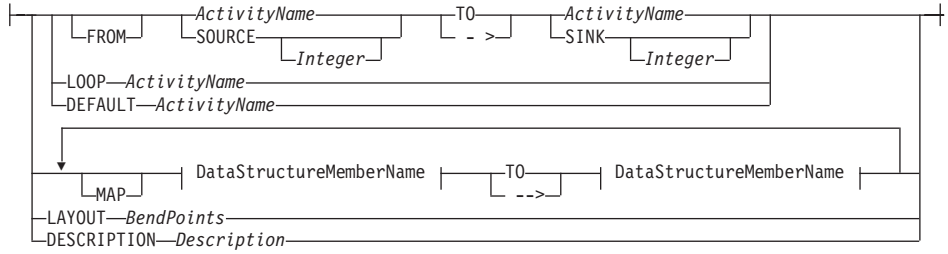
Control flow

Control Setting:

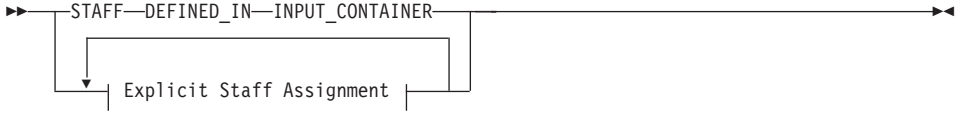


Data flow

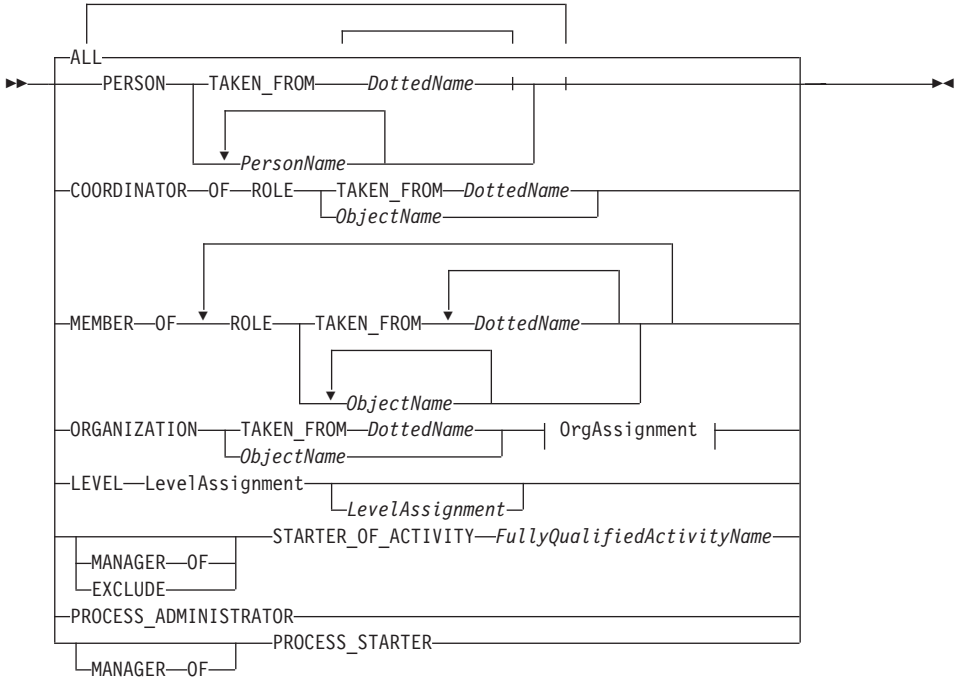
Dataflow Setting:



Staff Assignment



Explicit Staff Assignment

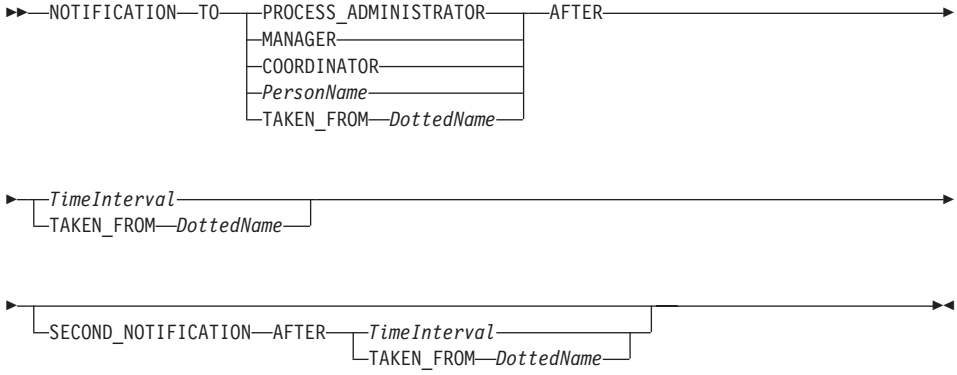


OrgAssignment:

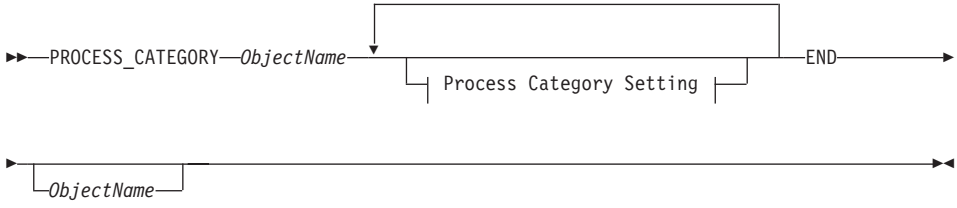
LevelAssignment:

Notification

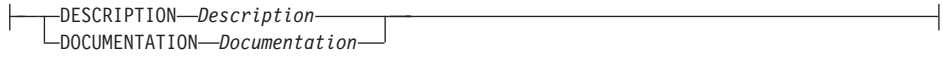
Explicit Notification



Process category

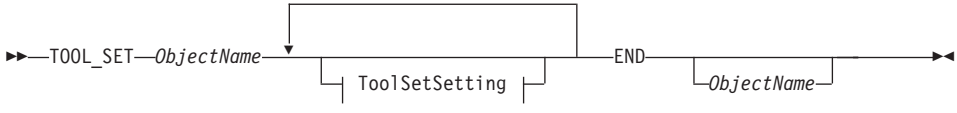


Process Category Setting:

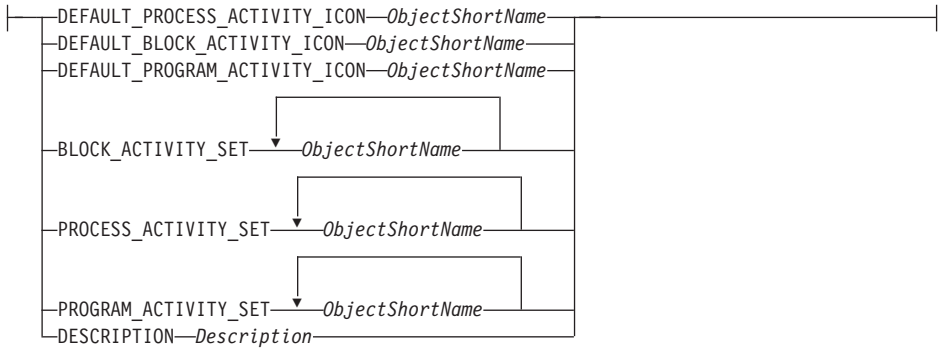


ToolSet

ToolSet is only valid for MQ Workflow Buildtime.



ToolSetSetting:



The following default values apply:

```
TOOL_SET 'STANDARD'
   DEFAULT_PROCESS_ACTIVITY_ICON 'fmcbprca'
   DEFAULT_BLOCK_ACTIVITY_ICON 'fmcbblka'
   DEFAULT_PROGRAM_ACTIVITY_ICON 'fmcbprga'
END 'STANDARD'
```

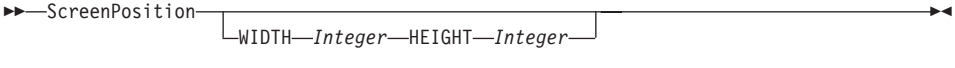
Common Variables

ScreenPosition

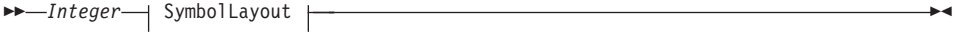
Notes:

1 Integer represents 0,1 mm.

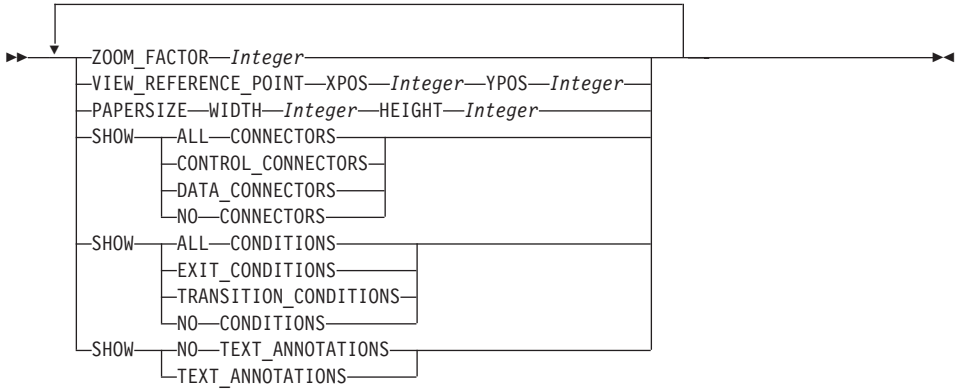
SymbolLayout



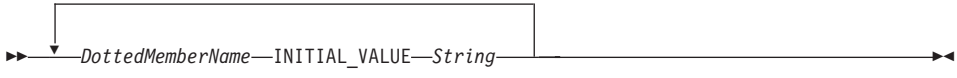
ContainerLayout



WindowLayout



ContainerInitial



BendPoints

Color

RedPart:

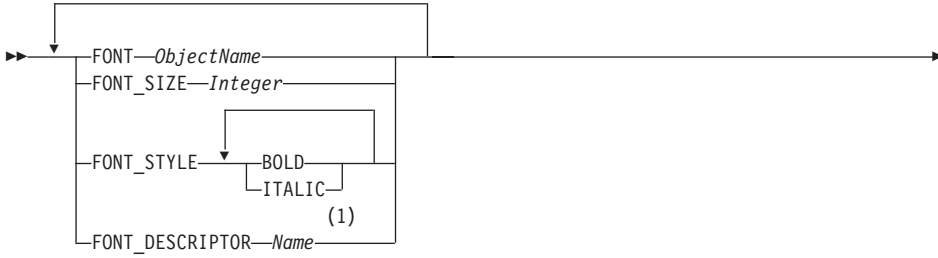
GreenPart:

BluePart:

ColorSetting

TextSettings

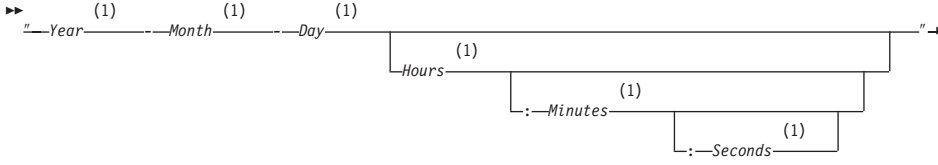
FontSettings



Notes:

1 The FONT_DESCRIPTOR is a platform-specific setting and contains additional information, such as the character set that you use for Windows NT/9x.

TimeStamp



Notes:

1 Integer, specifies UTC time. For example: `-1999-06-18 12:29:05`

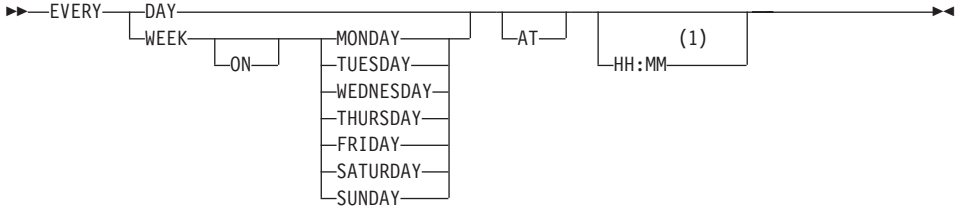
TimeInterval

Notes:

1 You can use YEARS and MONTHS only for "Process Staff Assignment Setting" on page 95 (Explicit process staff assignment setting – NOTIFICATION AFTER) and for "Notification" on page 100 (Explicit Notification TO).

TimePeriod

TimeEvent



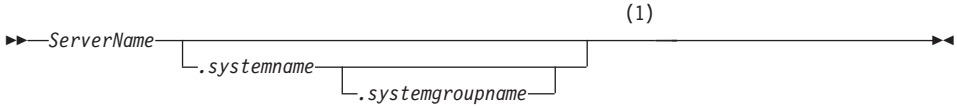
Notes:

1 Integer, specifies local time.

MessageLength

The minimum length is 256 KB and the maximum is 96 MB.

FullyQualifiedServerName



Notes:

1 ServerName, systemname, systemgroupname are Object Short Names. The defaults for systemname and systemgroupname are taken from the profile, which is generated during installation.

Part 3. Appendixes

Appendix A. Reorganizing your Buildtime database

Just like other relational databases, the Buildtime database must be reorganized on a regular basis to reduce its size. This helps to ensure that you do not run out of disk space.

When you are using a relational database, the amount of disk space increases when you add entries and the disk space is not reduced even if you delete entries. Therefore, you have to compact the database to gain new space and reduce the size of the database.

Before you start to reorganize or compact your database, back up your database with the tools that you use for your regular backup procedure. The file name of the Buildtime database is **fmcbtdb.mdb** and is stored in the directory that you specified during the installation for MQ Workflow.

Buildtime and IBM DB2 Universal Database

If you are using a DB2 (R) database for Buildtime, you can reorganize your database as described in the DB2 Administration Guide. The administration tools in the DB2 folder, which are part of your DB2 installation, are used to reorganize a DB2 database.

Using Microsoft Jet database engine

If you are using a Microsoft Jet database engine, you can use the **ODBC Data Source Administrator**, which is part of your installation.

To start the **ODBC Data Source Administrator**:

1. Open the **Control Panel**
2. Double-click **ODBC**

   If you cannot find the **ODBC** icon, check if the ODBCAD32.EXE exists in the system directory of Windows NT or Windows 95. This program is installed automatically with your Buildtime installation. Start the **ODBCAD32.EXE**.

   This opens the **ODBC Data Source Administrator**.
3. Look for the database name **fmcbtdb** in the **System DSN** tab or the **User DSN** tab and click the name of the database **fmcbtdb**
4. Click **Configure**

This opens the **ODBC Microsoft Access 97 Setup**. The correct database name displays automatically in this dialog box and in the subsequent dialog boxes.

5. Click **Compact**
6. Click **OK** in the **Database To Compact From** dialog box
7. Click **OK** in the **Database To Compact Into** dialog box

   A warning message displays: The database already exists. Do you want to replace it?
8. Click **Yes**

   An information message displays: The database was successfully compacted.
9. Click **OK** to finish reorganizing your database

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Deutschland
Informationssysteme GmbH
Department 3982
Pascalstrasse 100
70569 Stuttgart
Germany

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these

names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp.1993, 1999. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

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

- AIX
- DB2
- DB2 Universal Database
- FlowMark
- IBM
- MQSeries
- OS/2
- OS/390
- RISC System/6000

Lotus Notes is a registered trademark, and Domino and Lotus Go Webserver are trademarks of Lotus Development Corporation.

Microsoft, Windows, Windows NT and the Windows logo are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

This glossary defines important terms and abbreviations used in this publication. If you do not find the term you are looking for, refer to the index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

A

administration server. The MQ Workflow component that performs administration functions within an MQ Workflow system. Functions include starting and stopping of the MQ Workflow system, performing error management, and participating in administrative functions for a system group.

activity. One of the steps that make up a process model. This can be a program activity, process activity, or block activity.

activity information member. A predefined data structure member associated with the operating characteristics of an activity.

API. Application Programming Interface.

application programming interface. An interface provided by the MQ Workflow workflow manager that enables programs to request services from the MQ Workflow workflow manager. The services are provided synchronously.

audit trail. A relational table in the database that contains an entry for each major event during execution of a process instance.

authorization. The attributes of a user's staff definition that determine the user's level of authority in MQ Workflow. The system administrator is allowed to perform all functions.

B

bend point. A point at which a connector starts, ends, or changes direction.

block activity. A composite activity that consists of a group of activities, which can be connected with control and data connectors. A block activity is used to implement a Do-Until loop; all activities within the block activity are processed until the exit condition of the block activity evaluates to true. See also *composite activity.*

Buildtime. An MQ Workflow component with a graphical user interface for creating and maintaining workflow models, administering resources, and the system network definitions.

C

cardinality. (1) An attribute of a relationship that describes the membership quantity. There are four types of cardinality: One-to-one, one-to-many, many-to-many, and many-to-one. (2) The number of rows in a database table or the number of different values in a column of a database table.

child organization. An organization within the hierarchy of administrative units of an enterprise that has a parent organization. Each child organization can have one parent organization and several child organizations. The parent is one level above in the hierarchy. Contrast with *parent organization*.

cleanup server. The MQ Workflow component that physically deletes information in the MQ Workflow Runtime database, which had only been deleted logically.

composite activity. An activity which is composed of other activities. Composite activities are block activities and bundle activities.

container API. An MQ Workflow API that allows programs executing under the control of MQ Workflow to obtain data from the input and output container of the activity and to store data in the output container of the activity.

control connector. Defines the potential flow of control between two nodes in the process. The actual flow of control is determined at run time based on the truth value of the transition conditions associated with the control connector.

coordinator. A predefined role that is automatically assigned to the person designated to coordinate a role.

D

data connector. Defines the flow of data between containers.

data container. Storage for the input and output data of an activity or process. See *input container* and *output container*.

data mapping. Specifies, for a data connector, which fields from the associated source container are mapped to which fields in the associated target container.

data structure. A named entity that consists of a set of data structure members. Input and output containers are defined by reference to a data structure and adopt the layout of the referenced data structure type.

data structure member. One of the variables of which a data structure is composed.

default control connector. The graphical representation of a standard control connector, shown in the process diagram. Control flows along this connector if no other control path is valid.

domain. A set of MQ Workflow system groups which have the same meta-model, share the same staff information, and topology information. Communication between the components in the domain is via message queuing.

dynamic staff assignment. A method of assigning staff to an activity by specifying criteria such as role, organization, or level. When an activity is ready, the users who meet the selection criteria receive the activity to be worked on. See also *level*, *organization*, *process administrator*, and *role*.

E

end activity. An activity that has no outgoing control connector.

execution server. The MQ Workflow component that performs the processing of process instances at runtime.

exit condition. A logical expression that specifies whether an activity is complete.

export. An MQ Workflow utility program for retrieving information from the MQ Workflow database and making it available in MQ Workflow Definition Language (FDL) or HTML format. Contrast with *import*.

F

fixed member. A predefined data structure member that provides information about the current activity. The value of a fixed member is set by the MQ Workflow workflow manager.

(FDL) MQ Workflow Definition Language. The language used to exchange MQ Workflow information between MQ Workflow system groups. The language is used by the import and export function of MQ Workflow and contains the workflow definitions for staff, programs, data structures, and topology. This allows non-MQ Workflow components to interact with MQ Workflow. See also *export* and *import*.

fork activity. An activity that is the source of multiple control connectors.

form. In Lotus Notes, a form controls how you enter information into Lotus Notes and how that information is displayed and printed.

formula. In Lotus Notes, a mathematical expression that is used, for example, to select documents from a database or to calculate values for display.

fully-qualified name. A qualified name that is complete; that is, one that includes all names in the hierarchical sequence above the structure member to which the name refers, as well as the name of the member itself.

I

import. An MQ Workflow utility program that accepts information in the MQ Workflow definition language (FDL) format and places it in an MQ Workflow database. Contrast with *export*.

input container. Storage for data used as input to an activity or process. See also *source* and *data mapping*.

L

level. A number from 0 through 9 that is assigned to each person in an MQ Workflow database. The person who defines staff in Buildtime can assign a meaning to these numbers such as rank and experience. Level is one of the criteria that can be used to dynamically assign activities to people.

local user. Identifies a user during staff resolution whose home server is in the same system group as the originating process.

local subprocess. A subprocess that is processed in the same MQ Workflow system group as the originating process.

logical expression. An expression composed of operators and operands that, when evaluated, gives a result of true, false, or an integer. (Nonzero integers are equivalent to false.) See also *exit condition* and *transition condition*.

M

manager. A predefined role that is automatically assigned to the person who is defined as head of an organization.

message queuing. A communication technique that uses asynchronous messages for communication between software components.

N

navigation. Movement from a completed activity to subsequent activities in a process. The paths followed are determined by control connectors, their associated transition conditions, and by the start conditions of activities. See also *control connector*, *exit condition*, *transition condition*, and *start condition*.

node. (1) The generic name for activities within a process diagram. (2) The operating system image that hosts MQ Workflow systems.

notification. An MQ Workflow facility that can notify a designated person when a process or activity is not completed within the specified time.

notification work item. A work item that represents an activity or process notification.

O

organization. An administrative unit of an enterprise. Organization is one of the criteria that can be used to dynamically assign activities to people. See *child organization* and *parent organization*.

output container. Storage for data produced by an activity or process for use by other activities or for evaluation of conditions. See also *sink*.

P

parent organization. An organization within the hierarchy of administrative units of an enterprise that has one or more child organizations. A child

is one level below its parent in the hierarchy. Contrast with child *child organization*.

parent process. A process instance that contains the process activity which started the process as a subprocess.

pattern activity. A single and simple activity in a bundle activity from which multiple instances, called pattern activity instances, are created at run time.

person (pl. people). A member of staff in an enterprise who has been defined in the MQ Workflow database.

predefined data structure member. A data structure member predefined by MQ Workflow and used for communication between user applications and MQ Workflow Runtime.

process. Synonymously used for a process model and a process instance. The actual meaning is typically derived from the context.

process activity. An activity that is part of a process model. When a process activity is executed, an instance of the process model is created and executed.

process administrator. A person who is the administrator for a particular process instance. The administrator is authorized to perform all operations on a process instance. The administrator is also the target for staff resolution and notification.

process category. An attribute that a process modeler can specify for a process model to limit the set of users who are authorized to perform functions on the appropriate process instances.

process definition. Synonym for *process model*.

process diagram. A graphical representation of a process that shows the properties of a process model.

process instance. An instance of a process to be executed in MQ Workflow Runtime.

process instance list. A set of process instances that are selected and sorted according to user-defined criteria.

process instance monitor. An MQ Workflow client component that shows the state of a particular process instance graphically.

process management. The MQ Workflow Runtime tasks associated with process instances. These consist of creating, starting, suspending, resuming, terminating, restarting, and deleting process instances.

process model. A set of processes represented in a process model. The processes are represented in graphical form in the process diagram. The process model contains the definitions for staff, programs, and data structures associated with the activities of the process. After having translated the process model into a process template, the process template can be executed over and over again. *Workflow model* and *process definition* are synonyms.

process monitor API. An application programming interface that allows applications to implement the functions of a process instance monitor.

process-relevant data. Data that is used to control the sequence of activities in a process instance.

process status. The status of a process instance.

process template. A fixed form of a process model from which process instances can be created. It is the translated form in MQ Workflow Runtime. See also *process instance*.

process template list. A set of process templates that have been selected and sorted according to user-defined criteria.

program. A computer-based application that serves as the implementation of a program activity or as a support tool. Program activities reference executable programs using the logical

names associated with the programs in MQ Workflow program registrations. See also *program registration*.

program activity. An activity that is executed by a registered program. Starting this activity invokes the program. Contrast with *process activity*.

program execution agent. The MQ Workflow component that manages the implementations of program activities, such as .EXE and .DLL files.

program registration. Registering a program in MQ Workflow so that sufficient information is available for managing the program when it is executed by MQ Workflow.

R

role. A responsibility that is defined for staff members. Role is one of the criteria that can be used to dynamically assign activities to people.

S

scheduling server. The MQ Workflow component that schedules actions based on time events, such as resuming suspended work items, or detecting overdue processes.

server. The servers that make up an MQ Workflow system are called Execution Server, Administration Server, Scheduling server, and Cleanup Server.

sink. The symbol that represents the output container of a process or a block activity.

source. The symbol that represents the input container of a process or a block activity.

specific resource assignment. A method of assigning resources to processes or activities by specifying their user IDs.

standard client. The MQ Workflow component, which enables creation and control of process instances, working with worklists and work items, and manipulation of personal data of the logged-on user.

start activity. An activity that has no incoming control connector.

start condition. The condition that determines whether an activity with incoming control connectors can start after all of the incoming control connectors are evaluated.

subprocess. A process instance that is started by a process activity.

substitute. The person to whom an activity is automatically transferred when the person to whom the activity was originally assigned is declared as absent.

support tool. A program that end users can start from their worklists in the MQ Workflow Client to help complete an activity.

symbolic reference. A reference to a specific data item, the process name, or activity name in the description text of activities or in the command-line parameters of program registrations. Symbolic references are expressed as pairs of percent signs (%) that enclose the fully-qualified name of a data item, or either of the keywords _PROCESS or _ACTIVITY.

system. The smallest MQ Workflow unit within an MQ Workflow domain. It consists of a set of the MQ Workflow servers.

system group. A set of MQ Workflow systems that share the same database.

system administrator. (1) A predefined role that conveys all authorizations and that can be assigned to exactly one person in an MQ Workflow system. (2) The person at a computer installation who designs, controls, and manages the use of the computer system.

T

top-level process. A process instance that is not a subprocess and is started from a user's process instance list or from an application program.

transition condition. A logical expression associated with a conditional control connector. If

specified, it must be true for control to flow along the associated control connector. See also *control connector*.

translate. The action that converts a process model into a Runtime process template.

U

user ID. An alphanumeric string that uniquely identifies an MQ Workflow user.

V

verify. The action that checks a process model for completeness.

W

workflow. The sequence of activities performed in accordance with the business processes of an enterprise.

Workflow Management Coalition (WfMC). A non-profit organization of vendors and users of workflow management systems. The Coalition's mission is to promote workflow standards for workflow management systems to allow interoperability between different implementations.

workflow model. Synonym for *process model*.

work item. Representation of work to be done in the context of an activity in a process instance.

work item set of a user. All work items assigned to a user.

worklist. A list of work items assigned to a user and retrieved from a workflow management system.

worklist view. List of work items and notifications selected from a work item set of a user according to filter criteria which are an attribute of a worklist. It can be sorted according to sort criteria if specified for this worklist.

Bibliography

To order any of the following publications, contact your IBM representative or IBM branch office.

MQ Workflow publications

This section lists the publications included in the MQSeries Workflow library.

- *IBM MQSeries Workflow: List of Workstation Server Processor Groups*, GH12-6357, lists the processor groups for MQ Workflow.
- *IBM MQSeries Workflow: Concepts and Architecture*, GH12-6285, explains the basic concepts of MQ Workflow. It also describes the architecture of MQ Workflow and how the components fit together.
- *IBM MQSeries Workflow: Getting Started with Buildtime*, SH12-6286, describes how to use Buildtime of MQ Workflow.
- *IBM MQSeries Workflow: Getting Started with Runtime*, SH12-6287, describes how to get started with the Client.
- *IBM MQSeries Workflow: Programming Guide*, SH12-6291, explains the application programming interfaces (APIs).
- *IBM MQSeries Workflow: Installation Guide*, SH12-6288, contains information and procedures for installing and customizing MQ Workflow.
- *IBM MQSeries Workflow: Administration Guide*, SH12-6289, explains how to administer an MQ Workflow system.

Related publications

- *Frank Leymann, Dieter Roller, "Workflow-based Applications", IBM Systems Journal 36, no. 1(1997): 102–123*, you can also refer to the Internet: http://www.almaden.ibm.com/journal/sj361/leymann.html
- *Workflow Handbook 1997, published in association with WfMC*, edited by Peter Lawrence

Index

E

F

H

I

M

N

O

P

R

S

T

U

V

W

Readers' Comments — We'd Like to Hear from You

IBM MQSeries Workflow
Getting Started with Buildtime
Version 3.2

Publication No. SH12-6286-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SH12-6286-03

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 0446
Postfach 1380
71003 Boeblingen
Germany

Fold and Tape | **Please do not staple** | Fold and Tape

SH12-6286-03

Cut or Fold
Along Line

Part Number: 22L4281
Program Number: 5697-FM3

Printed in Denmark by IBM Danmark A/S

22L4281

SH12-6286-03

Spine information:

IBM MQSeries Workflow

Getting Started with Buildtime

Version 3.2