WebSphere® Business Integrator

Run Time

Version 2.1

WebSphere® Business Integrator

Run Time

Version 2.1

Note

Before using this information and the products it supports, read the information in “Notices” on page 171

First Edition (June 2001)

This edition applies to Version 2.1 of the IBM® WebSphere® Business Integrator (program number 5724-A78) and to all subsequent release and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this book

The *WebSphere Business Integrator Run Time* book provides the information necessary to understand and administer the Business Integrator run time system. The book provides you with information about:

- The business process managers and the gateways of Business Integrator
- Solution deployment procedures
- The administration of the run time system, including the use of the Business Integrator consoles.
- Troubleshooting and problem determination.
- Reference information about configuration fields.
- A list of external application programming interfaces.

To complement the *WebSphere Business Integrator Run Time* book, online help for the administration consoles provides you with assistance while you manage Business Integrator and the deployed solutions.

Who should read this book

This book is for administrators responsible for deploying solutions to the Business Integrator run time system, and for monitoring and managing the run time system and solutions. This book is also intended for those who perform any rquired initial problem determination.

What you need to know

You should have a solid understanding of Business Integrator, which you should have gained by reading the *WebSphere Business Integrator Concepts and Planning* book.

For the deployment and especially the undeployment of solution packages, it is recommended that you have considerable experience in the configuration and administration of the base products of Business Integrator.

Before you implement WebSphere Business Integrator

WebSphere Business Integrator uses multiple underlying products and technologies to support the solutions that you create and run. In general, before you implement Business Integrator, you will need to understand the underlying products and technologies that support your solution.

Before you implement Business Integrator, you or other members of your organization will need to be generally skilled in the activities listed below for similar solutions, products and underlying products and technologies. If you and other members of your organization do not possess these skills, you will need to obtain assistance, from qualified services staff, either from IBM or from third parties, to implement Business Integrator. You must be prepared to use the documentation of the underlying products and technologies. (This documentation is provided with Business Integrator or otherwise from IBM.)

When you plan, install, and configure Business Integrator, you will need to understand how to install and configure soem of the underlying products and technologies that you use in your installation. Business Integrator provides the installation of most of the underlying products and technologies into its run time environment. However, you might need to install and configure certain underlying products separately into either the build time or run time environment. You might also need to diagnose and correct installation problems with underlying products and technologies.

Before you design, develop and publish solutions, you will need to be:
- Generally familiar with system integration techniques in a business environment.
- Prepared to use the tools of the underlying products and technologies that your solution requires.
- Familiar with the run time behavior of the underlying products and technologies that your solution requires.
- Familiar with modeling concepts and techniques such as Unified Modeling Language, and related tools, with state machine concepts, and with visual flow composition-modeling concepts and techniques.
- Familiar with Internet and Electronic Data Interchange (EDI) concepts and technologies, if required by your solution.
- Prepared to research the existing applications, systems, and networks that you integrate with Business Integrator.
  – Inside your enterprise, they can be known as legacy systems, back-end systems, enterprise applications, or endpoint applications.
  – Outside your enterprise, they can be known as trading networks, private EDI networks, or similar networks that your solution requires.

Before you deploy, run, manage, diagnose, and tune Business Integrator, you will need to be prepared to use the management, trace, audit, exception handling, diagnostic and related tools of the underlying products and technologies that support your solution. You will need to be prepared to understand the solution itself to the degree needed for these tasks.

Conventions and terminology used in this book

The "Glossary of Terms and Abbreviations" on page 175 introduces the terminology relevant to Business Integrator.

How to send your comments

IBM welcomes your comments. You can send your comments by any one of the following methods:

1. Electronically to this address:

   ```
   idrcf@hursley.ibm.com
   ```

   Be sure to include your network address if you want a reply.
2. By FAX, to the following numbers:

   ```
   UK: 01962-842327
   Other countries: +44-1962-842327
   ```
3. By mail to the following address:

   User Technologies
   Mail Point 095
   IBM United Kingdom Laboratories
   Hursley Park
   Winchester
   Hampshire
   SO21 2JN
   United Kingdom

Chapter 1. Introduction

This book tells you what you need to know about the run time system of Business Integrator. It explains how the run time components of Business Integrator work, and provides you with the information you need to administer the run time system. It also provides you with guidance on troubleshooting problems that you might encounter.

This book contains the following sections:

- **"Chapter 2. Business Integrator run time environment" on page 3.**

  This chapter describes in detail the business process managers and the gateways of Business Integrator:
  - Business Flow Manager, the component that controls the flow of business processes.
  - Information Delivery Manager, the component that performs routing, transformation, and delivery of messages within Business Integrator.
  - Interaction Manager, the component that provides personalized access, that is, based on the role of the user, to the content and activities required by the business processes and applications.
  - Solution Manager, the component that provides the infrastructure for monitoring and managing the Business Integrator system.
  - Trust and Access Manager, the component that controls access to the system and its associated business applications.
  - Partner Agreement Manager, which implements the public process and trading partner agreements.
  - DataInterchange, which provides an EDI channel for message transformation and transport to and from EDI connected business partners.

  For each of these, their role in the run time system is discussed and functional differences in the business process managers between the Entry configuration and Enterprise configuration of Business Integrator are noted. The chapter also contains figures illustrating the flow of data, and the interaction of the business process managers, gateways, and other components within the run time system.
- **"Chapter 3. Deploying solutions" on page 35.**

  This chapter describes how a solution package is deployed to the run time system. It discusses how solution packages are constructed, what the different stages in deployment are, and how the Solution Deployment Wizard works.

- **"Chapter 4. General administration" on page 49.**

  This chapter describes the administrative procedures you can perform on the run time system. Most of these procedures are performed using the three consoles provided in Business Integrator.

  1. The Platform Console, which you use to monitor and manage the components of Business Integrator, and using the Solution Deployment Wizard, to deploy packages to the run time system.
  2. The Product Console Launchpad, which you use to monitor and manage the base products of Business Integrator, by launching their administrative consoles.
  3. The Solution Console, which you use to monitor and manage the exception, trace, and audit logs.

  Each of the consoles is provided with online help, which supplements the information in this book.

  The chapter also provides information about other administrative functions of Business Integrator, such as, starting and stopping Business Integrator services, defining new users, and archiving logged data.

- **"Chapter 5. Problem determination" on page 69.**

  This chapter describes how to use the log information provided by Business Integrator to diagnose problems. For the three types of log, that is, audit, trace, and exception; information is provided about how to control, filter, and interpret the log information, This chapter also describes how to analyse topology problems.

- **"Appendix A. Business Integrator LDAP" on page 77.**

  This appendix contains reference material for LDAP configuration fields.

- **"Appendix B. Business Integrator audit, trace, and exception configuration" on page 139.**

  This appendix contains reference material about the audit, trace, and exception handling services.

- **"Appendix C. External application programming interfaces" on page 165.**

  This appendix lists the Application Programming Interfaces (API) that are exposed for Business Integrator.

- **"Glossary of Terms and Abbreviations" on page 175.**

  This glossary defines all the terms that you need to understand the run time system of Business Integrator.

Chapter 2. Business Integrator run time environment

This chapter describes these Business Integrator run time components:
- "Interaction Manager".
- "Information Delivery Manager" on page 14.
- "Trust and Access Manager" on page 14.
- "Business Flow Manager" on page 20.
- "Solution Manager" on page 30.
- "Gateways" on page 31.
  – Partner Agreement Manager.
  – DataInterchange.

Interaction Manager

Interaction Manager provides a role-based desktop for access to the content and activities required by the business processes and applications. It supports the active collaboration of the business process activities with internal and external clients.

Deployment of Interaction Manager is dependent on customization to the particular requirements of the customer.

The Interaction Managers primary goal is to provide role-based content and services to the user's desktop. This is accomplished through:
- Views (JSPs)
- Controllers (Servlets)
- Models (Beans)

Interaction Manager is a framework that provides services and APIs that are used when developing solutions in a Model View Controller architecture. Interaction Manager also provides sample Servlets and JSPs. These are supplemented with user created Servlets and JSPs to customize the web interaction for specific solutions in order to support specific application views.

Interaction Manager provides:
- JSPs to provide the user interface to a Business Integrator solution.
- Servlets to handle logon, workflow, and access to Business Flow Manager.
- An inbox to notify users of new work items.
- A navigation framework to render role-specific, tree-structured links to access services that are hosted by the solution.

- Rules based filtering and rendering of content based on the user role in a solution.
- View support for creating and rendering adaptive documents that are accessed through the Business Flow Manager.
- An AServlet API with tooling support that is used to organize controller servlets and classes in a Model View Controller architecture.

Note: Interaction Manager provides device specific logon views. When a solution is developed, the solution must handle the rendering of information on different devices. For example, presentation on a personal computer is different than the presentation on a personal digital assistant.

Interaction Manager data flow

This section describes the sequences and the Business Integrator managers that are utilized to render role-based desktops on an end user's browser.

1. An end user logs onto an authorized solution by entering the URL for that solution in their browser.
2. WebSEAL Proxy Server authenticates the user through Policy Director to obtain the roles the user is authorized for in a given solution.
3. After successful authentication, the logon servlet in Interaction Manager performs authorization and basic initialization. Interaction Manager then directs the request to the solution entry point URL. Typically this is a servlet that performs the remaining initialization required for the particular solution and renders the solution desktop.
4. Next the logon servlet calls an Interaction Manager entry point URL that can be a servlet, JSP, or any valid URL that renders the user's desktop for one solution. Included are the list of services the user can access. Figure 1 on page 5 shows the sample desktop layout provided with Interaction Manager:

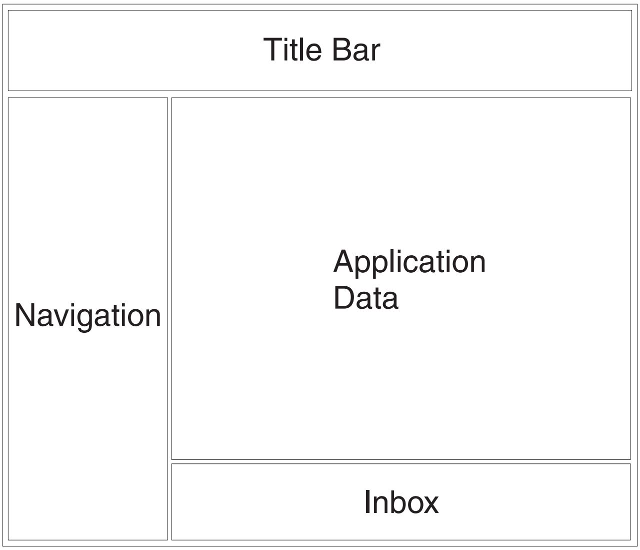


Figure 1. View rendered by Interaction Manager

In this suggested view, Interaction Manager renders the user’s desktop and places information in the frames as follows:

a. The title bar or masthead appears across the top of the browser window.
b. The navigation frame appears on the left side of the browser window beneath the masthead.
c. The application frame appears to the right of the navigation frame. This frame is the target frame for the navigation frame.
d. The inbox frame appears to the right of the application frame and beneath the application frame. This frame is optional and appears only if the logged on user is authorized for workflow services.

Figure 2 on page 6 shows a sample desktop as it could be rendered on a user’s desktop utilizing the type of default view provided by Interaction Manager.

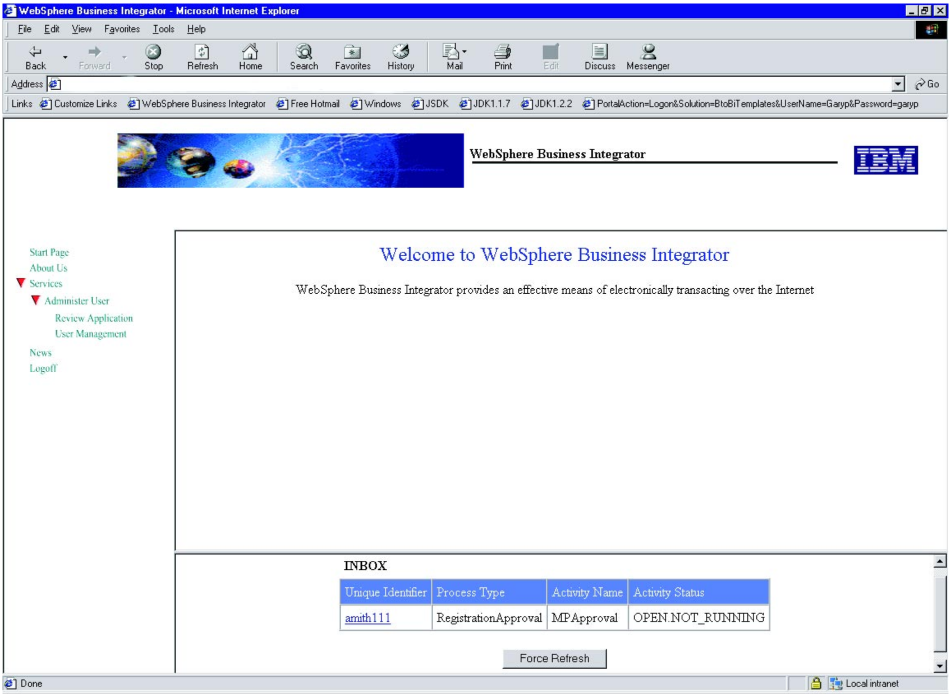


Figure 2. Sample Interaction Manager User Desktop (default view)

Navigation frame

The content of the navigation frame is based on the authorized roles of the user. A list of services the user can access is displayed in this frame.

Application frame

The content of the application frame is based on the authorized roles of the user.

The flow shown in Figure 3 on page 7 depicts the creation of a purchase order workflow instance from the application frame of a user's desktop.

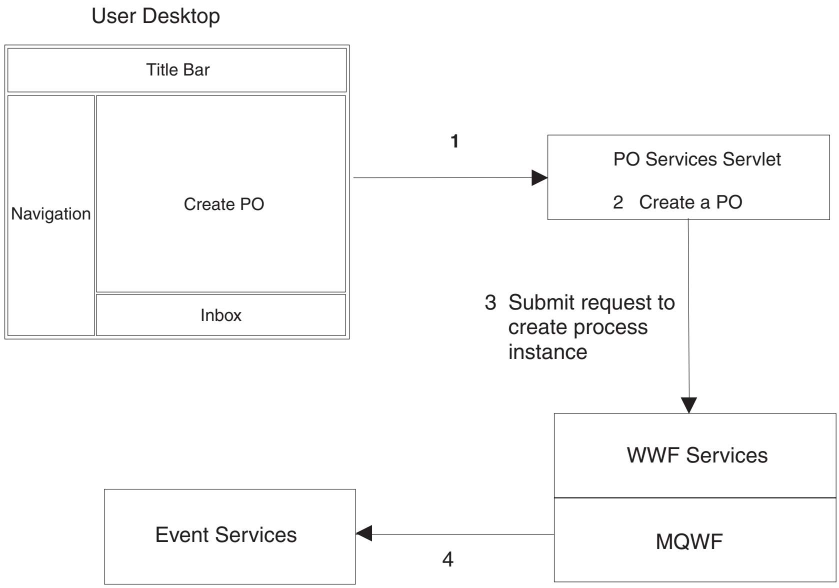


Figure 3. Create New Work Item through Interaction Manager

1. A user selects create a new purchase order from the application frame of their desktop. The request is sent to the URL of the purchase order servlet.
2. The servlet submits a request to WebSphere Workflow services and MQSeries Workflow to create a process instance of a purchase order.
3. MQSeries Workflow creates a process instance of the purchase order.
4. Event services is notified that a new purchase order process instance is available.

Inbox frame

This section describes how Interaction Manager controls the contents of the inbox frame on a user’s desktop. Three flow diagrams are provided that depict three unique usages of the inbox:

- An initial list of available work items when a user logs on to an application. See “Inbox Frame initial view”.
- Automatic update of available work items. See “Inbox frame automatic refresh” on page 8.
- Selection of available work items. See “Inbox frame user selects a work item” on page 9.

Inbox Frame initial view: The flow shown in Figure 4 on page 8 depicts how Interaction Manager initially renders a user’s desktop when the user first logs on to an application.

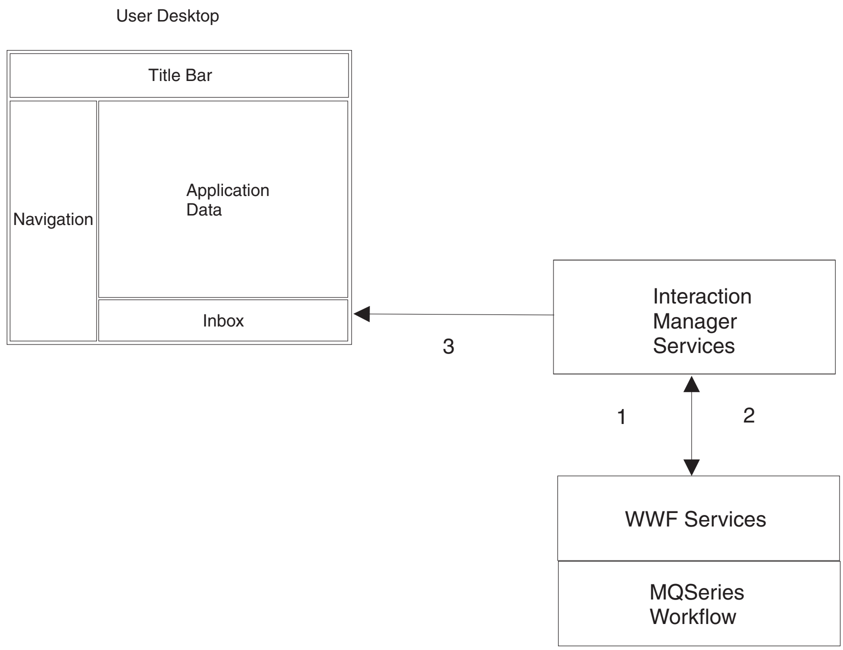


Figure 4. Interaction Manager renders initial view at user log on

1. Interaction Manager sends a query request to WebSphere Workflow Services and MQSeries Workflow.
2. MQSeries Workflow passes the desktop data back to Interaction Manager.
3. Interaction Manager renders the data on the user's desktop.

Inbox frame automatic refresh: The flow shown in Figure 5 on page 9 depicts how inbox data is automatically refreshed by Interaction Manager.

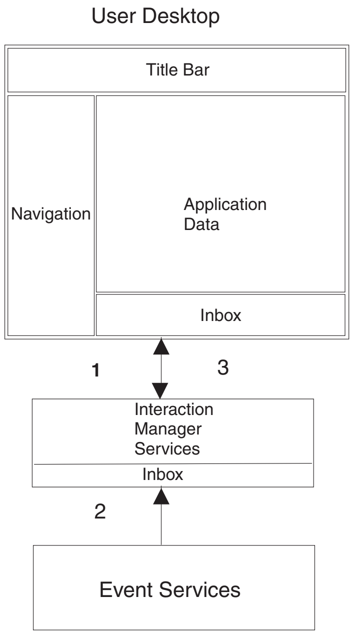


Figure 5. Interaction Manager automatic refresh of user inbox

1. The Interaction Manager JSP requests a refresh after a defined length of time specified in LDAP. The default interval is 60 seconds.
2. The Interaction Manager servlet subscribes and pulls events from event services.
3. Interaction Manager passes the data back to the user's desktop inbox. Any new events appear in the user's inbox.

Inbox frame user selects a work item: The flow shown in Figure 6 on page 10 depicts how data is rendered to the navigation frame when a user selects a work item from their inbox.

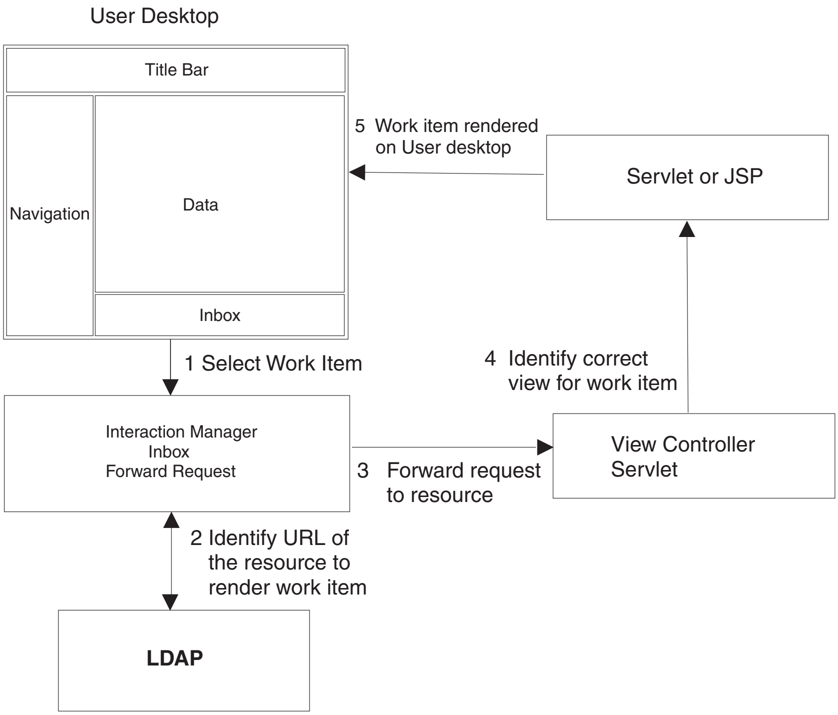


Figure 6. Select Work Item from the Desktop inbox

1. A user selects a work item from the inbox frame on their desktop.
2. The Interaction Manager inbox identifies the URL of the servlet that handles the work item.
3. Interaction manager forwards the work item request to the view controller servlet identified by the URL.
4. The view controller servlet determines the correct view for the requested work item and passes the data to the JSP or servlet that renders the user's desktop.
5. The servlet or JSP renders the desktop and passes the view to the application frame of the user's desktop.

Mitigating Simultaneous Work Item Access

Multiple users can access the same service, for example handling purchase orders. WorkFlow services provides a mechanism to prevent more than one user from accessing the same purchase order. For example:

1. User **A** claims purchase order 123 from their inbox.

2. The WorkFlow Servlet identifies the type of work item and forwards the request to the appropriate URL, for example Servlet **B**. Servlet **B** claims the work item and WWFServices publishes an event for the change in status of the work item.
3. The WorkFlow Servlet launches a polling sequence to refresh the desktops of all users that are authorized to claim the same purchase order. The polling interval is configured in LDAP. The default setting is 60 seconds.
4. Once the refresh is complete, other users that could have claimed PO 123 do not the PO in their inbox. It is no longer available and only the user that claimed the PO can work with it.

   Note: If another user attempts to claim the PO before the refresh completes, they should receive a message that the work item is no longer available. However, this must be implemented by the solution developer.

Adding a service to an existing solution

This section describes a method to add a new service hosted by Interaction Manager to an existing solution.

1. All services under a solution in Interaction Manager are registered in LDAP. For example, Solution Console and Purchase Order services. The services are registered by defining the menu structure for the service in LDAP.
2. For each solution, register menu entries in LDAP under the following hierarchy

```
o=ePIC
   o=ePICSolutions
      cn=[SolutionName]
        o=ePICMenuItem
           ePICMenuItemId=[unique menu item id]
```

3. On the Solution desktop viewed on the user's browser, all services are grouped under the default menu **Services**.
4. Group all menu items for a service under one root menu item. Use *ServicesMenu* for the new service's root menu in LDAP. Figure 7 on page 12 shows a typical grouping of services under a parent menu labeled "Services".

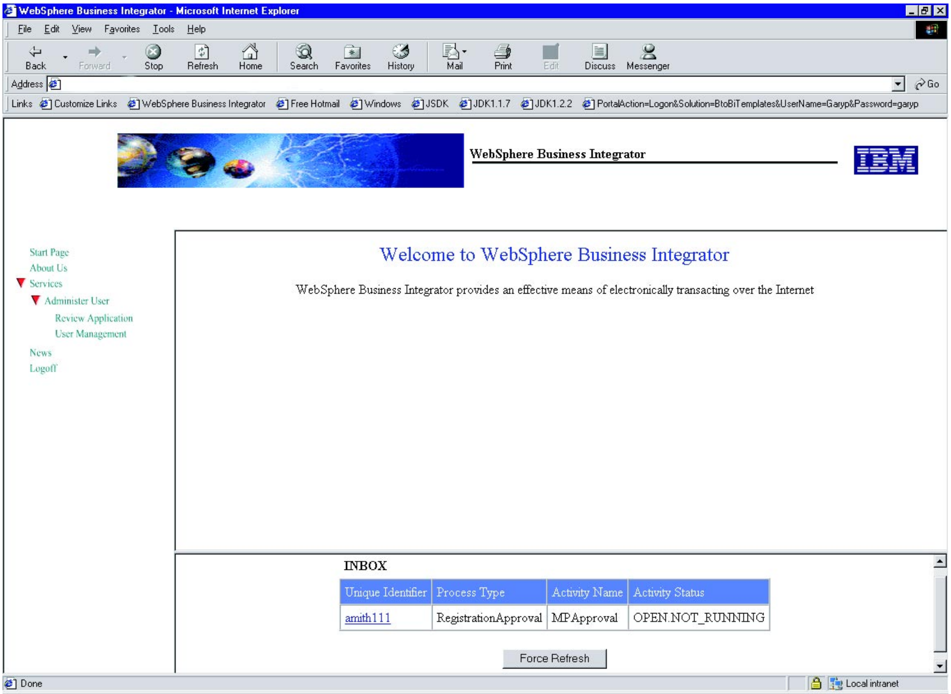


Figure 7. Sample Interaction Manager User Desktop Services Menu Structure

5. Define the menu tree structure under the service's root menu item. Figure 8 on page 13 shows a sample LDAP entry.

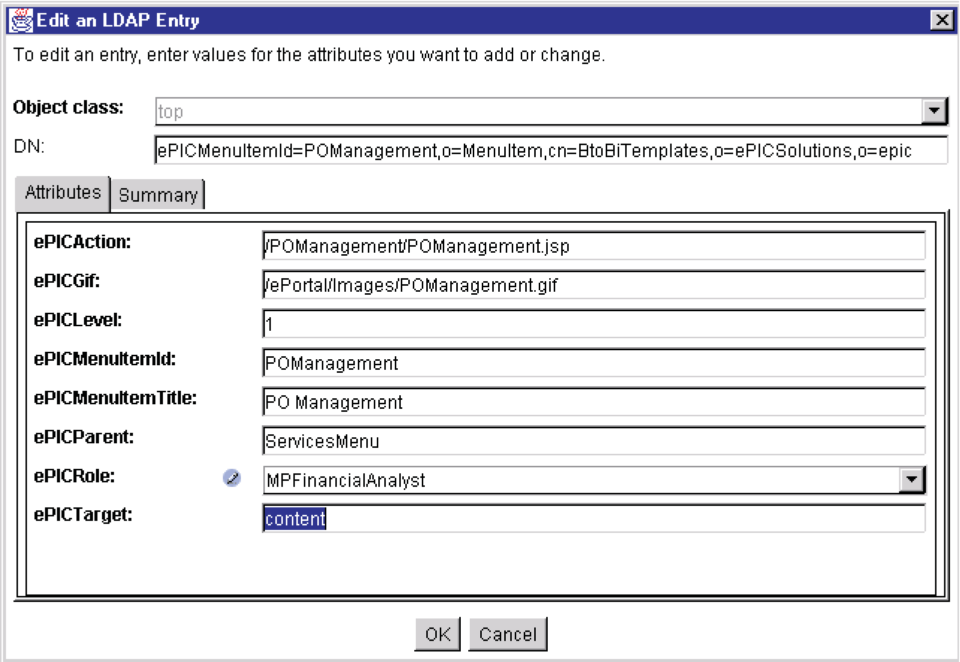


Figure 8. LDAP Interaction Manager Menu entry

6. Create image buttons for each menu item, and for each image assign a URL by which it can be accessed.
7. Define a target URL for each menu item. This is the URL that is invoked when a user selects a menu item. Interaction Manager renders the page in the application data frame of the user's desktop.
8. If required, create a new Web Application to support the new URLs using the WebSphere Administrators Console.
9. If the new service uses MQSeries Workflow, register the service in LDAP under: `o=ePIC,o=ePICSolutions,cn=[SolutionName],o=ePICProcessDefinitions, cn=[Process template name]`. This information is used by the inbox frame to render the selected work item.
10. Save the data, ensure that all program files, images and other support files are copied to the correct directories based on the web application's configuration.
11. Restart the application server where Interaction Manager is running. The new service is now available to authorized users.

Information Delivery Manager

Information Delivery Manager provides the messaging services for solutions built with Business Integrator Solution Studio. The services include message routing and delivery, transformation and tracing. These services are provided by MQSeries Integrator and MQSeries Adapter Offering's MQSeries Adapter Kernel and Adapter Builder. Information Delivery Manager is the messaging infrastructure for the run time environment.

MQSeries Adapter Kernel provides a run time infrastructure for the adapters providing:

- Message routing and delivery services through a Native Adapter.
- Hosting of adapters through the Adapter Daemon or on WebSphere Application Server within Business Integrator.
- XML Document services (Message construction).
- Tracing services.
- Separation of transport from communications message formatting.
- Dynamic communication transport ability.
- Dynamic communication message formatting ability.
- Usage of an XML configuration file or LDAP within Business Integrator.
- Additional communication transports:
  - MQSeries Server.
  - MQSeries Client.
  - Java Message Service.
- Java Development Kit support (Windows NT only)

For additional information see *MQSeries Adapter Kernel for Windows NT Quick Beginnings*, (GC34-5855). For a list of other MQSeries publications, see "Other Libraries" on page 193.

Trust and Access Manager

Trust and Access Manager provides security and directory services for business to business solutions created with Business Integrator. The primary focus of the Trust and Access Manager is to ensure that only authorized users and authorized applications can gain access.

Security

E-business solutions to business problems must consider security comprehensively to include enterprise-wide security policies, standards, and processes. Their development and deployment must ensure that the overall network is secure and appropriate mechanisms are in place to mitigate anticipated security risks.

For security, the Business Integrator run time environment is divided into three zones: the trusted zone, a demilitarized zone (DMZ), and the outside world. The trusted zone is a protected environment that ensures the integrity of data and allows processing of sensitive and classified information. This is where the business processing occurs. A DMZ lies between two firewalls and prevents direct access to the run time trusted zone. The outside world is a non-trusted zone. All clients enter the Business Integrator run time trusted zone from the outside world through the web.

Within the DMZ and between firewalls, a secure web proxy server provides the only exposed entry point for requests for services from web clients. The proxy server provides a junction point for authentication and authorization.

Except for DataInterchange, all other sotware products and managers in the run time environment use the Trust and Access Manager for security. Trust and Access Manager uses a mix of directory and security services to handle authentication, authorization and policy administration. It is also responsible for name resolution and directory access.

Trust and Access Manager uses the following products to fulfill security functions:
- IBM SecureWay Lightweight Directory Access Protocol (LDAP) Directory
- IBM SecureWay Policy Director

In addition, WebSphere security is enabled to provide security for WebSphere resources. WebSphere performs authentication and authorization to ensure that only authorized users can access EJBs.

Authentication overview
Business Integrator supports two approaches to authentication depending on the Business Integrator configuration. Business Integrator Entry uses only WebSphere security. Business Integrator Enterprise configuration extends WebSphere security by using Policy Director and optionally WebSEAL.

Authentication using WebSphere
Lightweight Third Party Authentication (LTPA) provides a framework to achieve global sign-on across a set of Web servers that fall within an Internet domain.

Business Integrator Entry Configuration security
This section describes how system security is implemented for Business Integrator Entry Configuration. Figure 9 on page 16 shows the sequence that occurs when a user attempts to access the system.

Run Time Environment

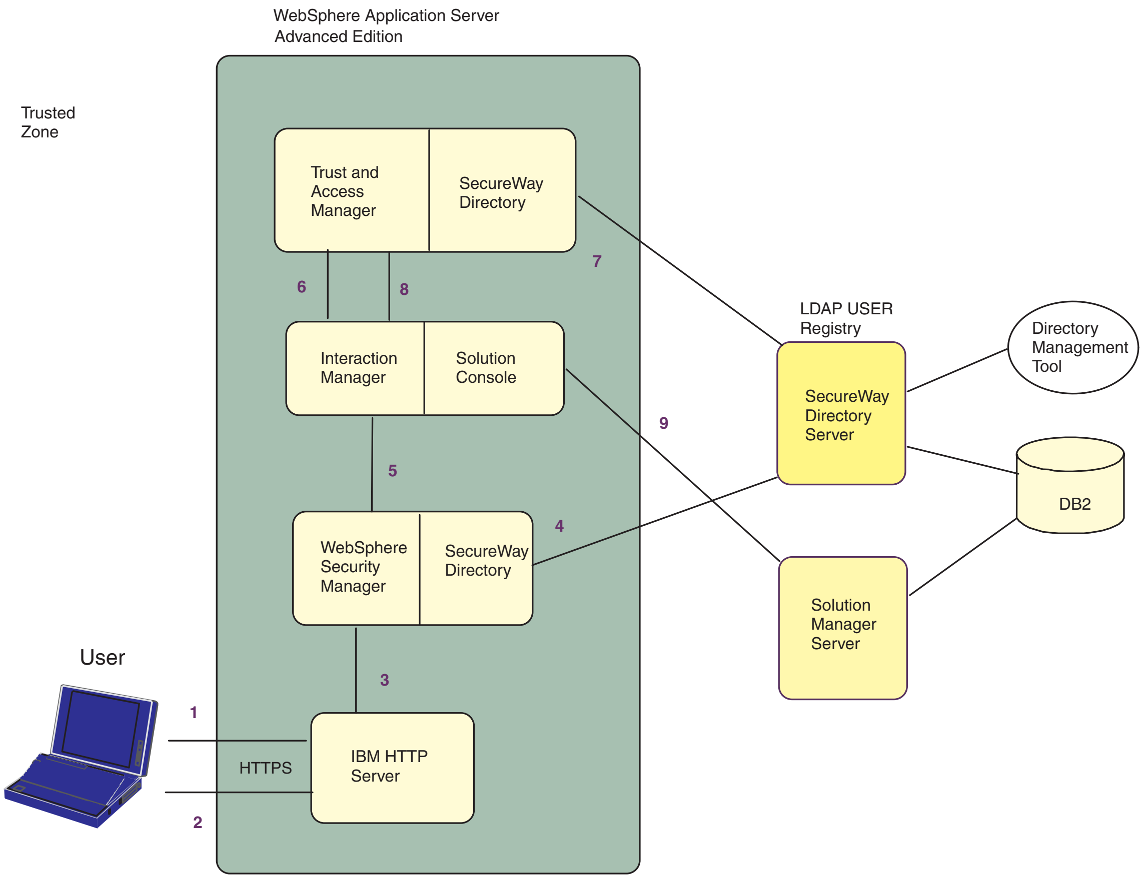


Figure 9. Security Flow Diagram Business Integrator Entry Configuration

1. A user enters a URL on their browser. An HTTP/SSL request is sent from the browser to the HTTP server.
2. The HTTP server sends its server certificate and issues an authentication challenge to the browser. The user submits their user ID and password.
3. The HTTP/SSL request is sent to WebSphere.
4. WebSphere accesses the LDAP user registry through the LDAP client to verify the userID and password.
5. If the userID and password are valid, WebSphere sends the HTTP/SSL request to Interaction Manager. Interaction Manager extracts the userID from the HTTP basic authentication header.
6. Interaction Manager contacts the Trust and Access Manager to obtain the authorized solution and roles for the user.
7. The Trust and Access Manager retrieves the solution and roles information for the user from LDAP.

8. The Trust and Access Manager returns an object that contains the solution and roles information to Interaction Manager. Interaction Manager uses the data in the object to render the users desktop.
9. If this user is authorized to access the Solution Manager, the Solution Manager Server handles requests from the Solution Console.

Business Integrator Enterprise Configuration security

This section describes how system security is implemented for Business Integrator Enterprise Configuration. Figure 10 shows the sequence that occurs when a user attempts to access the system.

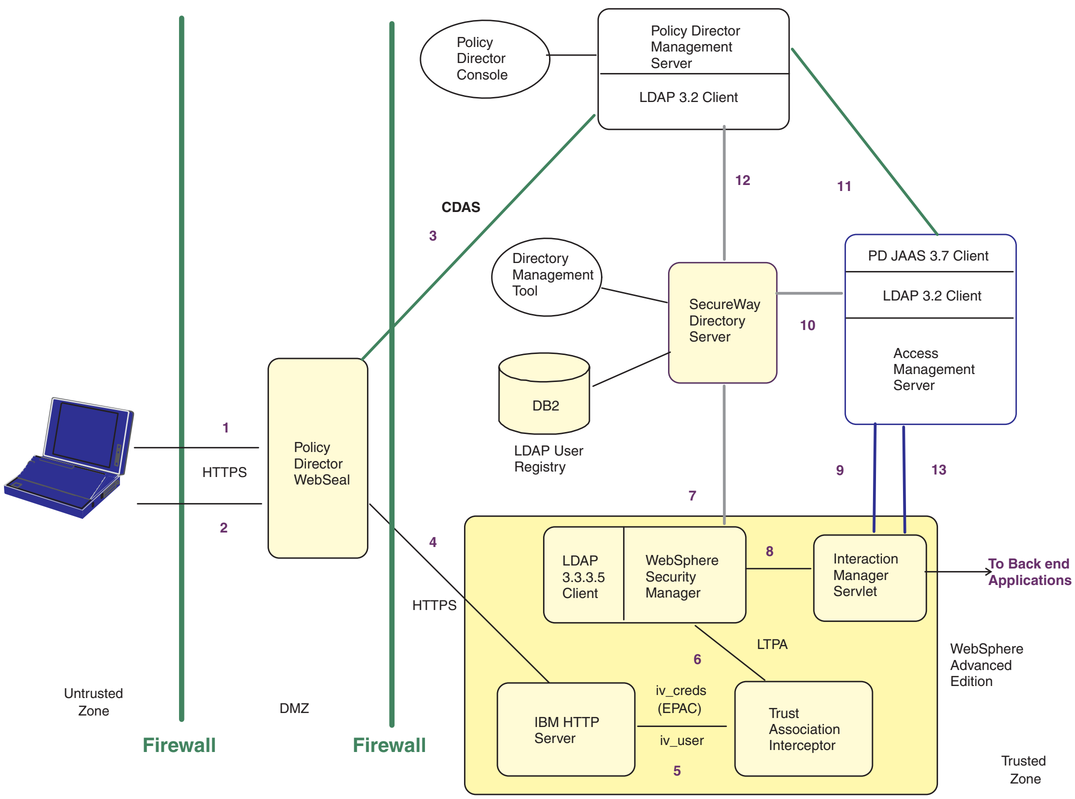


Figure 10. Security Flow Diagram Business Integrator Enterprise configuration

1. A client browser contacts WebSEAL using a Secure Sockets Layer (SSL) connection. WebSEAL intercepts the request and returns a signed certificate to the browser.
2. Once the browser accepts the server certificate, it returns a userID and password or a client certificate to WebSEAL.

3. WebSEAL sends an authentication request to Policy Director. Upon successful authentication, Policy Director creates an EPAC that WebSEAL caches for the duration of the Secure Sockets Layer session. Next WebSEAL generates an authorization request to verify that the user can access the URL that points to the JSP or servlet that is hosted by Interaction Manager
4. WebSEAL builds an HTTP request that contains the UserID (*iv_user*) and the Policy Director credential (*iv_cred*). The request is sent to the HTTP server.
5. The HTTP server forwards the HTTP request to WebSphere. The Trust Association Interceptor intercepts the request; processes the *iv_user* and *iv_cred* fields to create a WebSphere Application Server security context (LTPA).
6. The Trust Association then forwards the HTTP request to the WebSphere Security Manager.
7. The Security Manager accesses LDAP through Policy director to verify that the user can access the requested URL. Upon successful verification, the Security Manager sends the HTTP request to Interaction Manager.
8. Interaction Manager extracts the userID and password from the HTTP header.
9. Interaction Manager requests the authorized solution and roles for the user through the Access Management Server.
10. The Access Management Server checks the LDAP directory to retrieve the user solution and roles.
11. Then authorization requests for each solution and role combination are sent to Policy Director.
12. Policy Director accesses LDAP to obtain the requested information.
13. An object is sent to Interaction Manager that contains the authorized solution and roles, and Interaction Manager uses this information to render the user's browser.
14. Depending on which solution the user selects, Interaction Manager communicates with the back-end servers, such as the Business Flow Manager.

Note: The password that was entered by the user is not propagated by WebSEAL. Back-end processes retrieve the password from the Access Management Server.

Key Security Considerations

Business Integrator provides a security-related Application Programming Interface, the Access Manager Enterprise Beans. This API provides access to security functions that should only be accessible to a limited set of users. Generally System Administrators would be the typical users that should be authorized to access these security functions. As initially installed the Business Integrator security functions can be accessed by any user that has a valid WebSphere Application Server login and RMI/IIOP access to the WebSphere Application Server. IBM therefore strongly recommends that the following precautions are implemented:

- RMI/IIOP access is not allowed for users that access the system through firewalls.
- Either all users inside the firewalls in the trusted zone are trusted to be authorized for administrator access, or additional measures are taken in the deployed solution to deny access to the Access Manager Enterprise beans for users that are not authorized as administrators.

The Access Manager Enterprise Beans also provide Global Sign on capability, that enables password retrieval for use with GSO. This function can also be accessed by any user that has a valid WebSphere Application Server login and RMI/IIOP access to the server. In a typical solution it is desirable that the Global Sign on capability is provided for all authorized users, however, if Global Sign on is enabled for users within the firewall inside the trusted zone, those users must be trusted to be authorized for administrator access. For users that can not be authorized for administrative access, the Global Sign on capability must not be enabled.

Business Integrator security design assumes that all software that is installed on the application servers is trusted software. System Administrators must ensure that all deployed solutions are properly assessed for their security implications. To protect the application servers from the possible deployment of software that has not been assessed, ensure that access to Solution Manager consoles and applications is restricted to administrative users. See the *WebSphere Business Integrator Installation Guide* for details.

For any questions or concerns relating to security requirements, contact the Security Consultant for your enterprise or contact your IBM Representative.

Security audit logging

Business Integrator provides audit logging for security related activities. The ability to log and audit all access attempts is essential to secure the corporate Web. Monitoring access attempts by all users lets administrators detect security risks. SecureWay Policy Director centrally logs all access attempts using a standardized format and generates easy-to-read reports. This log is securely passed to DB2 for analysis of usage patterns.

Audit capabilities are provided by the base products:

- WebSEAL/Policy Director
- WebSphere
- DataInterchange

Non-repudiation

Non-repudiation prevents the sender of information from claiming, at a later date, that the information was never sent.

Non-repudiation logging is provided by Partner Agreement Manager for the receipt of trading partner messages.

Privacy/integrity

Integrity, known also as tamper detection, allows the recipient of information to verify and ensure that it has not been modified in transit. Any attempt to modify data or substitute a false message for a legitimate one can be detected.

Business Flow Manager

The Business Flow Manager consists of:

The base:

- WebSphere Messaging Services, the JMS Listener and associated worker message beans for asynchronous communication that uses MQSeries and MQSeries Adapter Kernel from endpoints and gateways. Worker message beans are a variant of the workers in MQSeries Adapter Kernel that are specialized for operation within WebSphere.

  The WebSphere Messaging Services enables applications and other clients to communicate to Business Flow Manager via messaging, in this case, MQSeries. It is addressed in WebSphere documentation.

  The JMS Listener invokes the worker message bean associated with the JMS destination queue from which the message was received. The JMS Listener passes the message to the worker message bean. (A worker message bean can be associated with more than one queue. Multiple message types can be received from the same queue.) The worker message bean calls the appropriate component within the Business Flow Manager.

  There is one set of MQSeries queues for at least each of the following:

  – Messages from the Partner Agreement Manager gateway.
  – Messages from the DataInterchange gateway.
  – User Process Execution Services (UPES) messages from MQSeries Workflow

  See "WebSphere Messaging Services" on page 21.
- Access to MQSeries Workflow:
  – WebSphere Workflow is an Object Management Group jointFlow compliant framework. It provides interfaces that are used for accessing the run-time environment of workflow engines, such as MQSeries Workflow. External application programming interfaces are not provided

for WebSphere Workflow. External interfaces are provided with WebSphere Workflow Services. See "WebSphere Workflow Services" on page 25.
- WebSphere Workflow Services, a wrapper around WebSphere Workflow. See "WebSphere Workflow Services" on page 25.
- Workflow collaboration enables a collaboration environment for MQSeries Workflow. See "Workflow collaboration support" on page 27.

- Solution Management services such as audit, exception handling, and monitoring of business processes. Solution Management is discussed elsewhere in this publication.

Utilities:

- Utilities are executable software and associated documentation that can prove useful in building and running solutions. See "Related documentation" on page 192for more information.

Part of the solution

- This is what you develop that runs in the Business Flow Manager. It provides the Business Flow Manager's adaptive document capability as well as other functionality depending on the needs of your solution.

WebSphere Messaging Services

WebSphere Messaging Services has several major uses:

- It receives JMS message datagrams in an asynchronous model.
- It receives a JMS message request and returns the reply in a synchronous request/reply model.
- It receives a JMS message datagram and returns the reply as a JMS message datagram.
- When an adapter throws an exception, it moves the JMS message to an error queue.

The basic process flow is:

1. JMS Listener receives a JMS message and passes it to a worker message bean.
2. The worker converts the JMS message into an MQAK holder object (EpicMessage class).
3. Based on the MQAK header values, the worker:
   a. Converts the application data into an appropriate input data structure.
   b. Determines the service session bean to invoke and passes the data object to a specific method on the service session bean.
   c. Converts any replies into a JMS message object and send the reply.
   d. In the case of error, puts the JMS Message object onto an error queue.

See *MQSeries Adapter Kernel for Multiplatforms: Quick Beginnings* for more information on JMS Listener and associated worker message beans. Worker message beans are a variant of the workers in MQSeries Adapter Kernel that are specialized for operation within WebSphere.

JMS side

The flow is:

1. The JMS listener is autostarted and runs within WebSphere Application Server.
2. It reads a configuration file to obtain configuration information relating the worker with the JMS destination.
3. When a JMS message is received by the JMS listener, the appropriate worker onMessage(JMSMessage) method is invoked.
4. The worker processes the message and sends any required replies.
5. If the worker encounters an error while processing the message, it puts the message onto an error queue.

Worker message bean: The message worker bean is a stateless session bean with the JMS Listener Interface ″onMessage(JMSMessage)″ implemented.

1. If after processing the message, there is reply data, the worker sends the reply.
2. While the message is being processed, if the worker receives an exception, the worker puts the original message on an error queue.
   a. If the message cannot be put to an error queue, the worker rolls back the transaction.
   b. If the worker rolls back the message, then the JMS Listener tries to reprocess the message up to a “retry” count. The default is a retry count of 0. If the retry count is reached, the JMS Listener logs the error and shuts down. This approach ensures that application data is not discarded.

Solution side

Once the worker onMessage( ) method has been invoked, the flow is:

1. Convert the JMS message object into an MQAK holder object.
2. Look up the service bean invocation parameters, based on the header values in the message.
   - Service session bean class to use.
   - Method to invoke in the service session bean.
   - Mapper class object to use for creating the data object required as a parameter in the method that maps between the message and the in terminal types.
3. Call the “InputMapper”.

4. Invoke the service session bean.
5. If there is reply data:
   - Call the "OutputMapper".
   - Send message.

Configuration

JMS Listener: The JMS Listener uses the JMS configuration file for worker message bean and JMS queue information. The name of the JMS configuration file is passed in as a property parameter to the Java command that is used to start the JMS Listener. For example,
`-DJMS_LISTENER_CONFIG_FILE=d:\wsjms\jmsconfig.xml`

Contents include:

JMS ConnectionFactory lookup name
Used for obtaining the JMS Connection Factory object.

JMS Destination (Queue) lookup name
Used for obtaining the JMS Destination Queue object.

Worker-MB home class lookup name
Used for obtaining the Worker Message bean.

Worker-MB remote class lookup name
Used for obtaining the Worker Message bean.

Maximum number of concurrent Worker-MB invocations
This will also determine if messages are processed serially or concurrently. A value of 1 means serially. No entry defaults to 1.

Transactional or not
Whether the JMS Listener retrieves the message under transaction control. No entry means Transactional.

Retry count
Indicates how many times to try reprocessing a message. No entry means 0.

Configuration information is required for each combination of JMS Destination Queue and worker.

The JMS objects are stored in WebSphere Application Server naming service (JNDI COS).

Worker message bean: Configuration information may be stored either in LDAP or the MQAK configuration file, aqmconfig.xml. Information is stored in a directory tree structure with primary information under the node BodyCategory ⇒BodyType.

Note: Because of the kernel behavior, default entries can exist that apply across all BodyCategories or all BodyTypes under a BodyCategory.

Under BodyType:

- Service Session Bean home reference.
- Method name to invoke on the Service Session Bean.
- Method parameter datatype. No entry will cause the use of the datatype of the returned object from the TerminalDataContainerMapper. Using a specific datatype name allows for those methods taking in super class type as its input parameter.
- Mapper class name to use.

Interfaces

Worker message bean
Generic code.

```
package com.ibm.epic.adapters.eak.adapterdaemon
```

TerminalDataContainer (TDC)
Represents the data and message type in object format for the service session bean method. A TerminalDataContainer class corresponds specifically to the input object type for a method in the service session bean. This class type is a concept rather than a specific class to extend or interface to implement. This avoids requiring MQAK for an enterprise bean client or browser that invokes the Service Session Bean. For this reason, no specific abstract class or interface is defined. However the class is expected to have get/setContext() methods as defined in this representative sample. This representative class is provided to those wishing to use it.

TerminalDataContainerMapper
TerminalDataContainerMapper is specific for a service session bean method TerminalDataContainer input object type and the XML string. The TerminalDataContainer class type is specific to the parameter type of the method. This is a helper class for the worker to use for converting the XML application data into the object representation the service session bean requires and back into an XML string. There is base class which is extended for dealing with specific application data types. This makes it easier for the worker to deal with a pre-defined class.

Service Session Bean
Is used for actually processing the data. It may represent an aggregate of message types, with each method type represented by a different method name and possibly a different DataObject.

ServiceEJBException

Common exception which can be thrown by the session service bean. By having a common exception, developers who write their code know what to expect. It is not specific to WebSphere in order to enable the code to be more application-server generic and to allow the session service beans to be invoked outside of WebSphere without requiring the WebSphere support jar files to be installed. The class is based on the MQAK AdapterException class.

Access to MQSeries Workflow

Access to MQSeries Workflow requires:

- WebSphere Workflow. See "WebSphere Workflow".
- WebSphere Workflow Services. See "WebSphere Workflow Services".
- Workflow collaboration support. See "Workflow collaboration support" on page 27.

WebSphere Workflow

Websphere Workflow consists of a set of enterprise beans that are hosted in WebSphere Application Servers. The interfaces for these beans:

- enable web-based and workflow aware applications to retrieve information about the status of workflows.
- associate resources with workflow objects.
- request execution.
- associate process data with workflow objects.

These beans do not have external application programming interfaces.

WebSphere Workflow Services

WebSphere Workflow Services is an enterprise bean that wraps the WebSphere Workflow enterprise beans, leveraging functionality by encapsulating blocks of invocations to the Workflow enterprise bean methods, in actions that meet the anticipated use patterns. Additionally, the WWFServices enterprise bean uses Business Integrator's trace and publishes events. These provide better integration with Solution Manager.

WebSphere Workflow Services leverages the functionality of WebSphere Workflow enterprise beans:

- Allows the same user to logon multiple times to MQSeries Workflow, unlike WebSphere Workflow.
- Provides a single method call to create a process instance, or claim, or to complete an activity, and so on.
- Publishes events for components such as Interaction Manager.

Signing on a user requires that WebSphere Workflow interacts with the Trust and Access Manager, to obtain the Global Sign On credentials for the user.

WebSphere Workflow Services are implemented as two enterprise session beans: WWFServices and WWFQuery. WWFQuery is a helper for WWFServices. WWFServices uses the exposed method on WWFQuery. No access bean is provided for WWFQuery and direct usage by the programmer is not supported.

The WWFServices bean's home and remote interfaces represent the set of functions exposed by the bean. Additionally, an access bean is packaged with the WWFServices bean to facilitate the client access. The access bean, WWFServicesAccessBean, provides the lazy instantiation of the session bean and exposes all remote methods of the WWFServices Session Bean. Access Bean construction is provided as part of the feature "IBM EJB Development Environment" of Visual Age for Java 3.5.

Configuration: WWFServices has the following attributes defined in LDAP under the AppId = WWFS.

EPICWWFEvents
: True means events are published. False means events are not published.

EPICWWFProfile
: True means profiling. False means no profiling.

EPICWASWorkflowProviderURL
: URL name where WebSphere Workflow is deployed.

EPICMQWFAdminUserId
: Admin userid for logon to MQSeries Workflow.

EPICMQWFAdminPassword
: Admin password for logon to MQSeries Workflow.

EPICMQWFGSOTargetResource
: Global Signon Target Name for MQSeries Workflow.

Global signon example: WWFServices gets the corresponding global signon userId and password for a given user through Trust and Access Manager using **GSO TargetName** stored in LDAP. For example, user RICHARD may have the GSO sign on userId for MQSeries Workflow as ″RICHARD″ and password as ″richard″. These user credentials are used to act on or fetch details from Websphere Workflow for the user RICHARD.

Publishing events: WWFServices publishes events based on the value for **EPICWWFEvents** defined in LDAP. WWFServices publishes events to the event server with the event type = ″WF″. The event header consists of event type, event topic and the role participating in that activity. The event topics are:

- AddActivity

- Claim
- Complete
- DeleteActivity

The event body consists of a string with following IDs. Their respective values are separated by ′=′ and each ID is separated by ′\n′:
- ACTIVITYNAME
- ACTIVITYSTATE
- ACTIVITYOWNER
- PROCESSNAME
- TEMPLATENAME

The role of the activity is fetched from LDAP. The use of Solution Studio to build the solution is required in order to maintain consistency between the MQSeries Workflow database and LDAP. When the solution is deployed, LDAP is updated with the activity and the role in which it participates.

Solution Studio works with these assumptions:
- Only role can be attached for an activity.
- Only one role will be attached for an activity.

If for some reason, WWFServices does not find an activity in LDAP, it publishes the events with the following event topics and with the role as ″UnknownRole″:
- AddActivity_NoRole
- Claim_NoRole
- Complete_NoRole
- DeleteActivity_NoRole

The event body is the same as above.

Workflow collaboration support

In order to operate with Business Integrator, MQSeries Workflow must have the capability to dynamically enable a collaboration environment. A workflow collaboration environment has these characteristics:
- Workflow activities that are part of the collaboration process are not known until run time.
- Multiple players participate in the process at the same time.
- Collaboration players and roles become known only during run time.
- It should be possible to enable a workflow collaboration environment at any point in the workflow graph.

Business Integrator provides the *collaboration program* (collab.jar). It enables workflow collaboration support in MQSeries Workflow. It is not required; you can choose to use it.

Collaboration program: The collaboration program is a Java bean. It runs on its own Java Virtual Machine. It is associated with an MQSeries Workflow activity. As input, it takes an XML string that is delimited with the start-tag `<Collab>`and the end-tag `</Collab>`. The XML string contains the roles, users and process templates (the workflow process definitions to be used for collaboration). For example:

```
<Collab>
<SPECIFICATION Type = 'Role' Value = 'Vendor' TemplateName= 'RFQProcess1' />
<SPECIFICATION Type = 'User' Value = 'JOHN' TemplateName= 'RFQProcess2' />
  <SPECIFICATION _ />
_
</Collab>
```

The following sequence describes how the collaboration program works:

1. The activity before the activity with which the collaboration program is associated must provide the XML string as part of a pre-defined container variable, `EPIC_COLLAB`, to make the XML string available for the collaboration program.
2. The activity associated with the collaboration program becomes active.
3. The collaboration program creates and starts processes based on the content of the XML string. For example, in the previous Collab XML sample it creates and starts:
   - An RFQProcess1 process for each of the users who can play the role of a Vendor.
   - An RFQProcess2 process, for the user ″JOHN″.
4. The collaboration program sends an event message for each process that it creates and starts. The event message has the properties described in *When the workflow process starts*.
5. The collaboration program waits for all the dynamically created processes to end before it recognizes that the workflow activity with which it is associated has completed. All the dynamically spawned processes must complete before the next activity in the workflow graph becomes available.
6. The collaboration program sends an event message when its associated activity completes. This event message has the properties described in the following list under *When the activity completes*.

When the workflow process starts

Event topic
WF

Event ID
CollabStart

Event body
“PROCESSNAME=processName ProcessId=processId PROCESSTEMPLATENAME=processTemplateName

PARENTNAME=parentName PARENTID=parentId
PARENTTEMPLATENAME=parentTemplateName”

where:

PROCESSNAME
Name of the process that is created by the collaboration program.

ProcessId
Object ID of the process that is created by the collaboration program.

PROCESSTEMPLATENAME
Name of the process template from which the process was created.

PARENTNAME
Name of the process that contains the activity that is associated with the collaboration program.

PARENTID
Object id of the process containing the activity that is associated with the collaboration program.

PARENTTEMPLATENAME
Name of the process template from which the process that contains the activity that is associated with the collaboration program was created.

When the activity completes

Event topic
WF

Event ID
CollabEnd

Event body
“PARENTNAME=parentName PARENTID=parentId
PARENTTEMPLATENAME=parentTemplateName”

See “Using the event service” on page 147 for information about event service.

The collaboration program requires:

- The MQSeries Workflow runtime user registry must be available outside the runtime engine. It should be stored in a file called *epicStaff.fdl*. This file can be generated using the following command:

```
fmcibie /e=c:\epicStaff.fdl /u=admin /p=password /c"EXPORT ROLE*"
```

- The user assigned to the activity that is associated with the collaboration program must be logged on to the workflow engine before starting the process containing the activity that is associated with the collaboration program. This can be accomplished using the following command:

  ```
  fmcxspea -u=username -p=password -f
  ```

  Where:

  username
  The name of the person assigned to the activity.

  password
  The password of the person assigned to the activity.

  You can chose to assign a different user to each activity or to assign one user to all the activities. In both cases, each user will need to log on.
- The files *fmcoutil.jar* and *b2biwf.jar* must be included in the system classpath.
- The WebSphere Business Integrator event service must be running.

Business Integrator provides a file, *bfmrobot.bat*, that can be used to perform the above steps with the exception of starting event service. You might need to modify this file before using it. You must run this file before creating and starting the process that contains the activity that is associated with the collaboration program.

Solution Manager

Solution Manager provides consoles, and services and applications which include:

- Console access to monitor and manage solutions, through audit and exception logging and tracing.
- An environment where Business Integrator, solutions, and underlying software components can be easily instrumented to measure performance.
- An interface to allow supported management applications to get and set attributes of managed objects and provide control of those objects.
- Access to all the managed objects of the system from a single point.
- Solution Manager is initialized based on information contained in the topology.

Solution services

Solution services provides the following:

- Reliable logging of trace, audit and exception data.
- Event service.
- Console functions to manage the run time services and to view and manage log contents.

By default, all log, trace, and exceptions are recorded by a logging service running on the Database and Directory Server that writes the records to the database. Based on a type field, the entries may be routed to different tables within the logging service. These tables can not be extended and should not be changed. Tables are provided for audit, trace, and exceptions. The log tables include the following columns:

- A column for each context field in the MQAK header.
- A column containing a subject in the context for application-provided correlation (for example, a Purchase order number).

The logging service uses MQSeries Adapter Offering running within WebSphere. This enables receipt and processing of multiple messages within one transaction. Instead of routing to the default logging service, messages may also be routed to a custom service (for example, an exception handler), based on their type (audit, trace, exception). Viewing the audit and exception logs is provided through the Solution Manager Console.

Gateways

Partner Agreement Manager

The Partner Agreement Manager gateway provides the capability for exchanges with trading partners over the Internet, using a variety of business protocols (channels) between the partners. Partner Agreement Manager therefore facilitates the integration of inter-enterprise business processes, as opposed to the intra-enterprise business processes that are integrated with the rest of the Business Integrator product suite.

The Partner Agreement Manager gateway is provided by these products:

- Partner Agreement Manager, which provides gateway connectivity, through the definition of public processes that are shared between partners, and the use of integration adapters and tooling. Tools that allow the creation of solution artifacts specific to the Partner Agreement Manager gateway can be launched from within Solution Studio.
- Partner Agreement Connect, which can be used as a limited-function alternative to Partner Agreement Manager in the Entry configuration. Partner Agreement Connect does not allow the definition of new public processes, and supports a limited number of trading partners. Partner Agreement Connect would be used as the spoke of a trading hub, where the hub owner wishes to control the public process definitions of its participating partners.
- Partner Agreement View, which enables the connection of Web applications to Partner Agreement Manager, to allow users to interact with public processes and business objects through the Internet.

The Partner Agreement Manager gateway can be managed from the Business Integrator Solution Manager, for example, to allow the configuration of trading partners at run time. For more information, see "Administration consoles of other products" on page 56.

For more information about the Partner Agreement Manager product, refer to "WebSphere Partner Agreement Manager library" on page 193.

Public processes and private processes

The public process, also known as the inter-enterprise process, is, the step-by-step flow of information and actions between two or more trading partners. The private process is a trading partner's internal sequence of actions for its steps in the public process.

You create public processes by using the PAM Public Process Editor, which you can launch from Solution Studio. Although you can define both public and private processes with the Public Process Editor, you usually define public processes only, as the private processes are autogenerated by a deployment plug-in when the solution is deployed using the Solution Deployment Wizard (see "Chapter 3. Deploying solutions" on page 35). However, there may be circumstances where you need to generate the private process steps without using the Solution Deployment Wizard. An example of this is when you are the partner who has received the public process from a remote partner. In this case, you would use the bizProcessDeploy tool to import the public process and generate the private process steps. For more information, see the *WebSphere Partner Agreement Manager Business Process Integration Adapter Guide* book.

For more information about creating and editing public and private processes, refer to the *WebSphere Studio Business Integrator Extensions Developer's Guide* book.

Channels

In Partner Agreement Manager, a channel is an encapsulation of all the processing information needed to send messages to a trading partner's system, and to translate data from a trading partner into Partner Agreement Manager messages. All Partner Agreement Manager installations have the PAM-to-PAM channel installed. Available non-Partner Agreement Manager channels include the RosettaNet and XML channel.

For channel security, software including Java Secure Socket Extension (JSSE) is used for certificate-based authentication and authorization. The repository for authentication credentials is the LDAP directory.

PAM Proxy Server

Access control on inbound messages from trading partners to Business Integrator and on outbound messages from Business Integrator to trading partners is performed by the PAM Proxy Server, which resides in the DMZ.

The PAM Proxy Server is a passive proxy, in that it requires the connection to be initiated from the trusted zone. It passes on the HTTP or other request unmodified, and without performing any authentication.

Business Process Integration Adapter

Business Integrator provides the Business Process Integration Adapter, which enables Partner Agreement Manager to work with the rest of Business Integrator. The Business Process Integration Adapter packages XML message payloads from a public process into message formats that can be transported by Information Delivery Manager to Endpoints or to the Business Flow Manager. It packages messages from Endpoints or the Business Flow Manager into message formats that can be sent through a public process. The Business Process Integration Adapter manages the correlation information that is necessary to bind public processes in Partner Agreement Manager with business processes in the Business Flow Manager.

For more information about Business Process Integration Adapter, refer to the *WebSphere Partner Agreement Manager Business Process Integration Adapter Guide*.

DataInterchange

Provides an EDI channel for message transformation and transport to and from EDI connected partners.

Three DI transports are:

- MQ EDI Services
- IBM IE
- IBM Base MQSeries

DataInterChange Features

DataInterchange includes the following features:

- Flexible setup and administration
  - Online customization of standards, maps, and trading partner relationships.
  - Mapping designed to support:
    - Literals/constants.
    - Accumulators, arithmetic and logical operations.
    - Qualified loop and element mapping.
    - Hierarchical loop mapping.
    - Envelope field mapping.
    - User exits at the field level.
    - User-defined translation and validation tables.

  - Boolean logic.
- Maps can be used by one or more trading partners.
- Export/import available to move user data between test and production systems.

- Superior translation capability
  - Syntax checking.
  - Test and production support.
  - Ability to translate and envelope separately.
  - Flexible command language interface.
  - Interactive, batch, event-driven, and real-time processing.
  - Automatic generation of functional acknowledgments.
- Versatile communications
  - Support for networks and direct connections to trading partners.
  - Ability to resend individual transactions or entire envelopes.
  - Support for MQSeries Queues as a means of exchanging data between trading partners.
- Extensive reporting and auditing
  - Reporting of trading partner relationships, including the transaction set being used, and the last communication with a trading partner, and others.
  - Reporting of envelope and transaction status for both online and batch processing.
  - Exception reporting.
  - Setting acceptable error levels for the trading partner/map combination.
  - Reporting of SAP status for online and batch processing.
  - Optional audit log with archive recovery capability.
- Standards support:
  - Multiple versions and releases of standards.
  - Electronic standards distribution to speed delivery of new standards.
  - Ability to migrate a map from one version/release of a standard to another, or from one transaction to another.
  - Online creation and customization of standards.
  - Full standards compliance checking (user option).

For more information about DataInterchange, refer to *WebSphere Business Integrator DataInterchange for Windows NT User's Guide*.

Deployment of DataInterchange artifacts

DataInterchange artifacts are not published using Solution Studio, and are therefore not included in solution packages. Instead, there is a manual process for the deployment of the artifacts.

For information about the deployment of DataInterchange artifacts, refer to the *WebSphere Business Integrator DataInterchange for Windows NT User's Guide*.

Chapter 3. Deploying solutions

This chapter describes how solutions are deployed in Business Integrator. It describes:

- What deployment is
- What a solution package is and what it contains
- What the different stages in deployment are
- How the Solution Deployment Wizard works
- How to undeploy a solution
- How to redeploy a solution package

Use this chapter to perform the following tasks:

Task	**Tools**	**Procedure**
Deploy a solution package created in Solution Studio to the run time system	Solution Deployment Wizard in Platform Console	See “Using the Solution Deployment Wizard to deploy the solution package” on page 42 and the online help.
Complete deployment of a partially deployed solution	Solution Deployment Wizard in Platform Console	See “Completing a partial deployment” on page 45
Undeploy a solution	Solution Deployment Wizard, bizRemoveSolution tool	See “Undeploying a solution” on page 45.
Redeploy a solution	Solution Deployment Wizard in Platform Console	See “Redeploying solution packages” on page 47 and the online help.

What is deployment?

During solution development, artifacts such as enterprise beans and Flow Definition Language (FDL) files are created and these must be installed, configured, and activated on the run time system. Deployment is the process of making solution elements (groups of artifacts) available to the run time system.

The solution elements to be deployed must first be packaged. The *packaging* process is done in Solution Studio and it involves creating a zipped solution package containing the elements to be deployed, together with metadata describing the elements, and scripts to automate the installation, configuration, and activation of the solution elements. The solution elements represent indivisible logical units of the solution that must be installed on a single machine.

On the run time system, you launch the Solution Deployment Wizard from the Platform Console to deploy the solution package. Using the metadata in the solution package, the solution elements are distributed to the correct machines, and are installed, configured, and activated by running the deployment scripts. As the Solution Deployment Wizard runs, it updates the status of the solution and its elements, in the Topology Repository.

To perform deployment, the Solution Deployment Wizard calls a deployment application that runs on top of a deployment framework that is installed with Business Integrator. The deployment application invokes the deployment scripts through the deployment framework, which contains various plug-ins on remote machines that perform the actual deployment processing.

The deployment plugins are Managed Beans (MBeans), which are part of the Java Management Extensions (JMX) architecture.

Solution packages

Solution packages are created during the packaging process of Solution Studio. Refer to the *WebSphere Studio Business Integrator Extensions Developer's Guide* book for details of how solution packages are created.

A solution package is a .zip file that contains:

- An XML file (metadata.xml) that describes the contents of the solution package, including its solution elements, their zip files and the run time components to which the elements belong.
- For each solution element, a separate .zip file containing:
  - All the artifact files for the solution element
  - XML deployment scripts, for installation and activation.
  - Any additional files that may be required by the deployment scripts

The metadata.xml file defines:

- The name and version of the solution.

  These are stored in the Topology Repository and will be shown in topology tree views, when you select **View**, then **Solution** in the Platform Console. (For a full discussion of topology, refer to the *WebSphere Business Integrator Concepts and Planning*.)
- For each element of the solution:
  - The name of the element.
  - The logical Business Integrator component (manager) and subcomponent with which the element is associated, for example, Business Flow Manager or Information Delivery Manager, and its subcomponent, for example, WebSphere Application Server or MQSeries Workflow. This hierarchical structure is used to build the solution view in the Topology Repository, and to determine the target machine for the element.
  - The name of the .zip file containing the artifacts, and any associated files.
  - The name of the XML script files for installation and activation.
  - The name of any text files that provide manual instructions for installation and activation.
  - The name of any files that contain symbols that need to be substituted to map logical to physical entities.

For more information about the metadata.xml file, refer to the *WebSphere Studio Business Integrator Extensions Developer's Guide* book.

Solution package contents

The following is an example of the contents of a solution package .zip file:

- metadata.xml
- bfm/process/flows.zip

- bfm/was/ejbs.zip
- wp/was/webserver.zip

For this example, the bfm/process/flows.zip file might contain:

- install.xml - installation script
- act.txt - manual activation instructions
- And other dependent files, such as abc/xyz.fdl, that are used by install.xml

The bfm/was/ejbs.zip file might contain:

- install.xml - installation script
- instructions.txt - manual instructions
- And other dependent files, such as abc/xyz.jar, abc/another_script.xml that are used by install.xml

The wp/was/webserver.zip file might contain:

- install.xml
- And other dependent files, such as abc/xyz.jsp, pqr/xyz.jar, ijk/xyz.html that are used by install.xml

The deployment process

After a solution is developed, it is ready to be published and then deployed on the run time system.

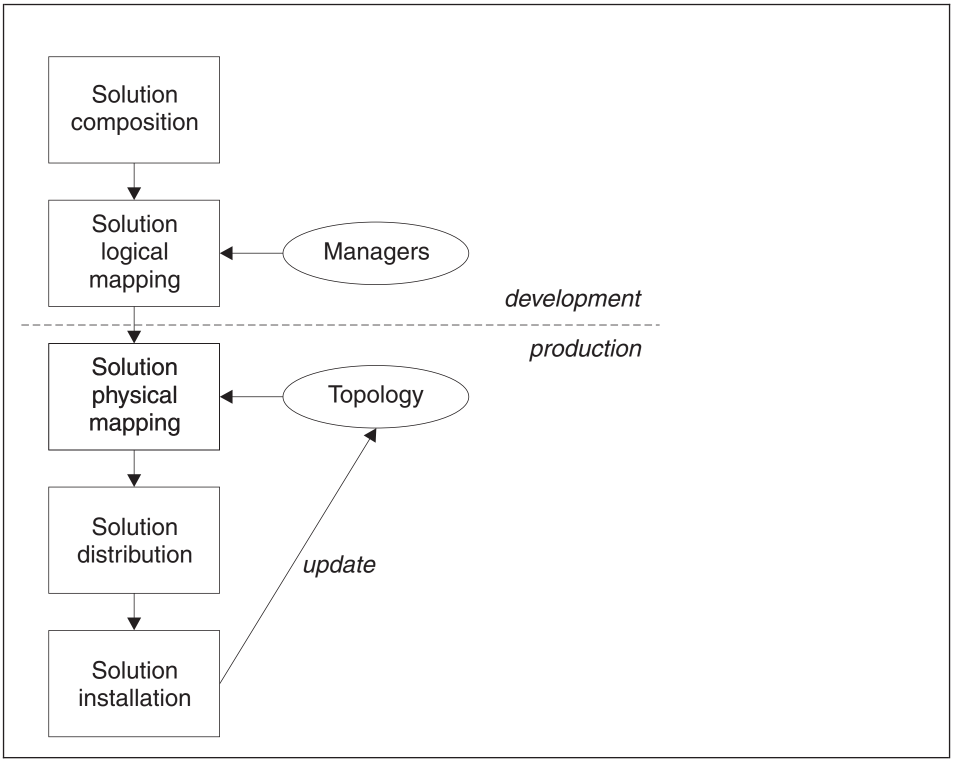


Figure 11. Overview of deployment

The overall process of packaging and deployment consists of the following steps (see also Figure 11).

1. **Solution logical mapping.**

   This step is done by the developer in Solution Studio and involves identifying the artifacts of the solution that need to be deployed, grouping them into solution elements, and mapping those elements to run time Business Integrator components. For example a solution element containing a group of enterprise beans could be mapped to Business Flow Manager, and its subcomponent WebSphere Application Server.

   The output of this step is a solution package that contains the elements of the solution, and for each element, associated metadata that is used in the run time stages of solution deployment.

Each solution element has associated scripts that drive the automatic installation and activation of the solution element on the target machine. When automatic installation or activation of a solution element is not possible for all of the stages of installation and configuration, there will instead be a text file containing instructions for the manual installation and activation required.

For more information, refer to the *WebSphere Studio Business Integrator Extensions Developer's Guide*.

2. **Accessing the solution package on the run time system**.

   You access the solution package from the machine that contains the Platform Console, that is, the machine containing the Solution Manager facility. For more information about the Platform Console, see "Monitoring and managing the Business Integrator platform" on page 56. You can copy the solution package from diskette, or access it from CD-ROM, or a mapped network drive.

   Subsequent steps are performed by the Solution Deployment Wizard.

3. **Storing the solution and its elements in the Topology Repository**

   You start the Platform Console and start the Solution Deployment Wizard by clicking first the **File** menu, then **Deploy Solution**. You can then select the solution package, and after clicking the **Next** button, the solution package is unpacked. After you click the **Start** button, the deployment application stores the data about the solution and its elements in the Topology Repository as the deployment steps are performed.

4. **Solution physical mapping.**

   For each solution element in the package, the Solution Deployment Wizard uses the metadata (component and subcomponent) of the solution element to identify from the Topology Repository the facility to which the solution element applies. The facility is then used to identify the physical target machine for the solution element.

   The deployment scripts may require information from the Topology Repository to operate correctly, for example, IP addresses of machines for other facilities. The Solution Deployment Wizard will therefore substitute symbolic names in script files with attributes from objects in the Topology Repository, or LDAP directory as necessary. After substitution, the modified solution element zip file is stored in the document root of the HTTP server, ready to be downloaded by the next phase of deployment.

5. **Solution distribution.**

   The deployment application instructs the deployment framework to download the solution element zip file from the HTTP server to a temporary directory on its target machine, ready for installation.

Before the distribution step begins, the status of the element in the Topology Repository is updated to `Distributing`. When a solution element has been successfully distributed, the status of the element is updated to `Distributed`.

6. **Solution installation.**

   The deployment application starts the installation process, and the plug-ins in the deployment framework run the scripts to install the elements on the target machine.

   Some solution elements cannot be installed automatically, and in this case a dialog box containing installation instructions is displayed. These instructions must be carried out before the deployment process can continue.

   Before the installation step begins, the status of the element in the Topology Repository is updated to `Installing`. When the installation step completes, the status of the element is updated to `Installed`.

   Note: No solution artifacts are identified in the solution metadata.

7. **Solution activation**

   The deployment application starts the activation process, and the plug-ins in the deployment framework run the scripts to activate the elements on the target machine.

   Some solution elements cannot be activated automatically, and in this case a dialog box containing activation instructions is displayed. These instructions must be carried out before the deployment process can continue.

   Before the activation step begins, the status of the element in the Topology Repository is updated to `Activating`. As each element is activated, the status of the element is updated to `Activated`.

   After all the solution elements have been activated, cleaning up takes place to remove temporary files on the target machines. As each element is cleaned up, the status of the element in the Topology Repository is updated to `Deployed`.

   When all elements have successfully activated, the status of the solution in the Topology Repository is updated to `Activated`.

Figure 12 on page 42 shows the different states of the solution and solution elements during the deployment process.

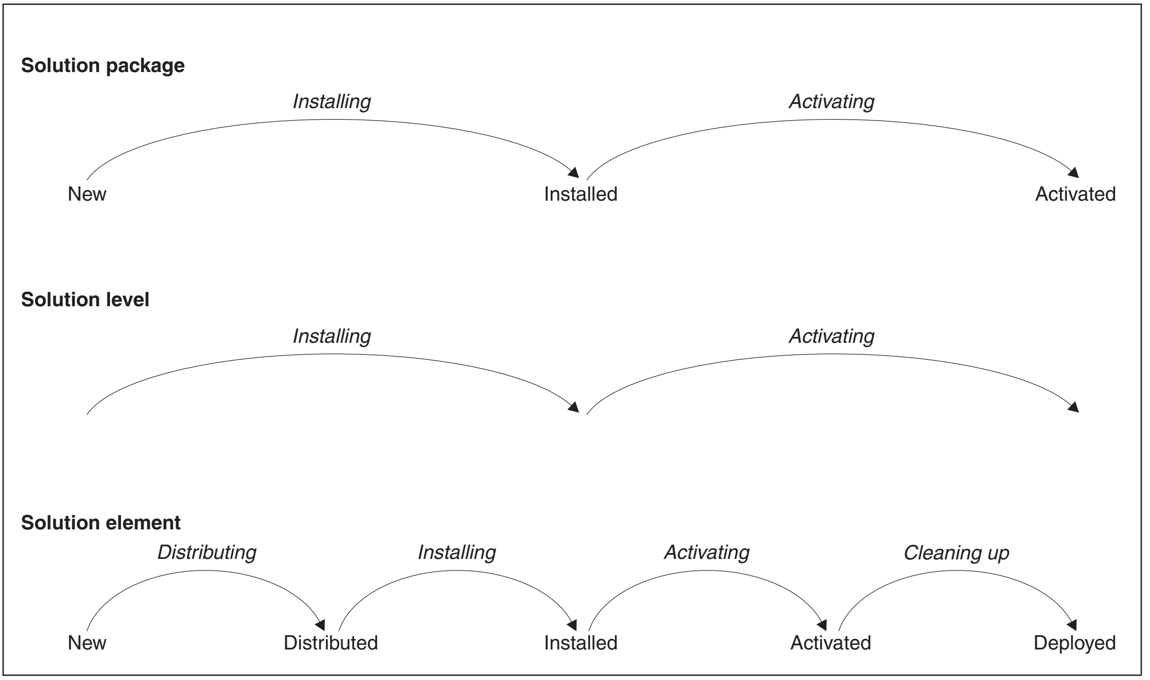


Figure 12. Deployment state transitions

Using the Solution Deployment Wizard to deploy the solution package

You can use the Solution Deployment Wizard to:

1. Deploy a solution package
2. Complete the deployment of a solution package that was partially deployed
3. Redeploy all or part of a solution package that was already deployed

Notes:

1. You cannot run multiple instances of the Solution Deployment Wizard concurrently.
2. If you are going to use the Solution Deployment Wizard to deploy a solution package containing WebSphere Application Server artifacts, you must first disable WebSphere security. For further information about enabling and disabling WebSphere security refer to the *WebSphere Business Integrator Installation Guide*.

To use the Solution Deployment Wizard:

1. Start the Platform Console.
2. Click the **File** menu and then **Deploy Solution**.

3. Enter the pathname of the solution package, and then click **Next**.

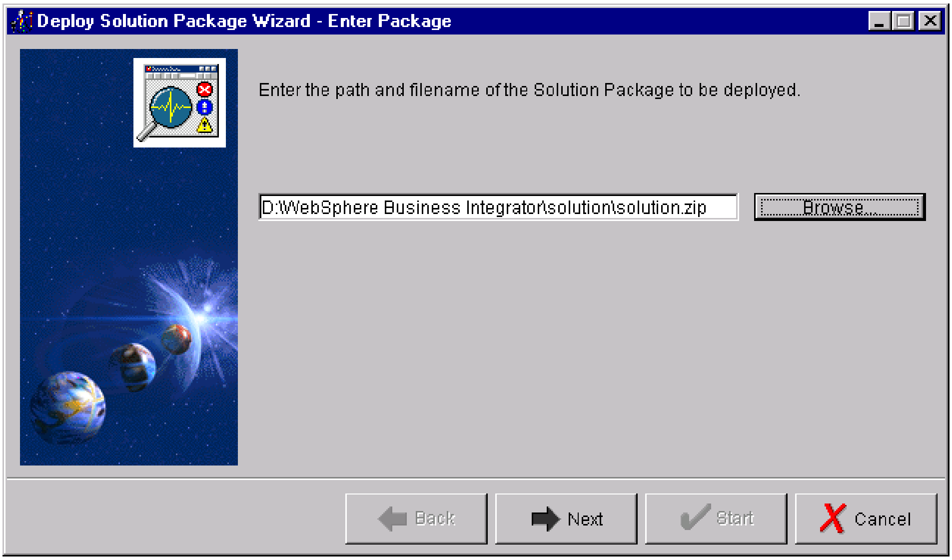


Figure 13. Solution Deployment Wizard

4. The next dialog box displays the details of the solution package that you selected.

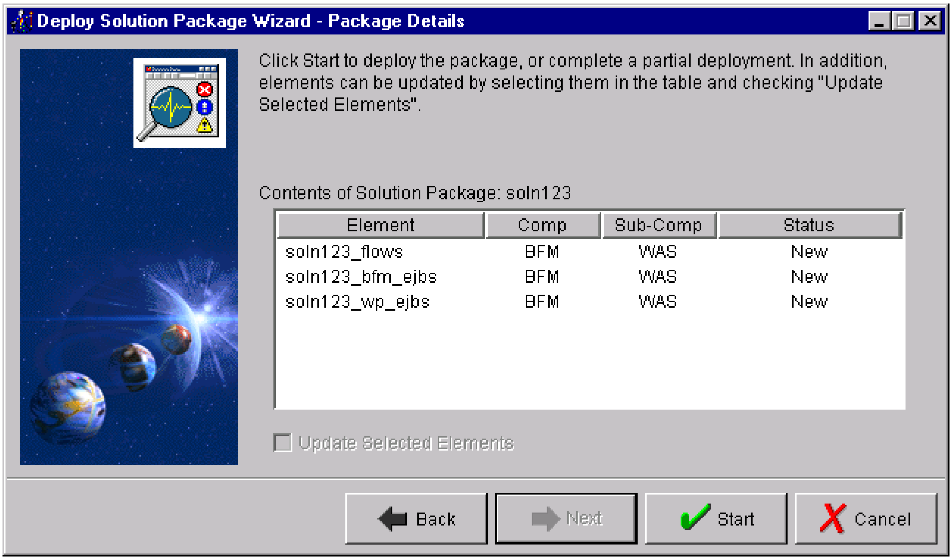


Figure 14. Solution Deployment Wizard

The manual instructions required for deploying the package are output to the *solutionname*_man.txt file in the \logs subdirectory where you installed Business Integrator. You may want to examine this file so that you know what is involved for the manual instructions that will be displayed later.

5. If you need to force the redeployment of solution elements that have already been deployed or partially deployed, select the required elements in the table and then select **Update Selected Elements**. You would force redeployment, for example, if problems with a package are discovered, and you need to redeploy certain elements. The selected elements will be deployed again as though they had never been deployed. Any elements that are not selected, and which have not been fully deployed will have their deployment completed.
6. If you are happy to continue with deploying this solution package, click **Start**, otherwise click **Back**.
7. A progress bar shows how much of the solution package has been deployed, and log messages are displayed to indicate which solution element is being deployed, and which stage this represents in the deployment.
8. In the case of solution elements that cannot be deployed automatically, a dialog box is displayed with instructions for the manual deployment of the element. You must either:
   a. Carry out the manual instructions, and click **Completed**, or,
   b. Click **Cancel Deployment** if you wish to carry out the instructions later. In this case, when you later restart deployment, the process will continue where it left off.
9. When the deployment is finished, a message is displayed and the Topology Repository status of the solution changes to `Activated`. Click **Finish**, to exit from the Solution Deployment Wizard.

After you have run the Solution Deployment Wizard, click **Update Status** and then **Solution** from the **View** menu, to display a topology tree including the deployed solution.

If deployment is not successful, messages are displayed informing you that errors occurred, or that deployment instructions for some artifacts could not be verified for success, and that these instructions need to be verified manually. You should check the deployment logs on the local machine to determine which artifacts encountered problems, and take appropriate action to verify the affected instructions manually See “Deployment log messages” on page 48.

For information about messages generated by the Solution Deployment Wizard, refer to the *WebSphere Business Integrator Messages* book.

Completing a partial deployment

The Solution Deployment Wizard allows you to complete the deployment of a solution that was interrupted.

Deployment may be interrupted due to:

- Failure of deployment. In this case, you should fix the problem that caused the failure (perhaps a network problem) before restarting the deployment. To help determine what the problem was, you should examine the deployment logs, see “Deployment log messages” on page 48.
- Clicking **Cancel Deployment** when presented with manual deployment instructions.

To complete the deployment of a partially deployed solution:

1. Start the Platform Console if necessary.
2. Click the **File** menu and then **Deploy Solution**.
3. Enter the pathname of the solution package, and then click **Next**.
4. The Package Details panel (see Figure 14 on page 43), shows the deployment status of the elements in the package. Do not select any of the elements and do not select **Update Selected Elements**.
5. To complete the deployment of the solution package, click **Start**.
6. Carry out any manual instructions, and click **Completed**, or click **Cancel Deployment** if you wish to complete deployment later, after performing the manual instructions.
7. When the deployment is finished, a message is displayed and the Topology Repository status of the solution changes to `Activated`. Click **Finish**, to exit from the Solution Deployment Wizard.

Undeploying a solution

Business Integrator does not provide automatic undeployment of solutions. To undeploy a solution you must therefore manually perform the reverse processes of distributing, installing, and activating of the solution elements. That is, you must deactivate, uninstall, and remove the solution elements. For deactivation and uninstallation, you need to know what actions were taken by the scripts run by the deployment plug-ins to perform installation and activation.

Using an undeployment wizard, you can unzip a solution package and generate a list of instructions for undeploying the solution. This is an HTML file containing instructions based on the structure of the solution package, and the status of any solution elements that have been fully or partially deployed.

In some cases, after you deploy a solution, you may deploy other solutions that depend on the first solution, that is, the solutions have elements in common, such as enterprise bean containers and queue definitions. In such cases, you must first undeploy the dependent solutions before you undeploy the first solution.

If you only have one solution, or if you are certain any dependent solutions have been removed, then you should:

1. Generate undeployment instructions from the solution package, see "Generating the undeployment instructions".
2. Remove the solution from the Topology Repository, using the bizRemoveSolution tool, see "Removing a solution from the Topology Repository" on page 48.
3. Perform the undeployment instructions.

Generating the undeployment instructions

To unzip a solution package and generate a list of instructions for undeploying the solution:

1. Click **Generate Undeployment Instructions** in the **File** menu in the Platform Console.
2. In the Undeploy Package panel, specify the path and filename of the solution package to be unzipped, and the name of the directory into which the solution package is to be unzipped.

The undeployment instructions file (undeploy.html) is generated in the *solution package name* subdirectory of the directory that you specify. The directory will contain subdirectories that reflect the structure of the solution package. As for deployment, symbolic names in script files are substituted with attributes from objects in the Topology Repository, or LDAP directory.

The unpacked and substituted solution elements are unzipped into the following directory structure:

```
undeploy directory\solution package name\element name
\directory within solution element ZIP
```

For example, if the specified directory is `c:\undeploy` and the solution package name is `UserRegs` and the solution element name is JMSLDAPMQ, then the solution element zip file is unzipped into c:\undeploy\UserRegs\JMSLDAPMQ.

Performing the undeployment instructions

The undeployment instructions file contains a section for each solution element, and within these sections separate sections for deactivation and uninstallation. You must follow the instructions in the order they appear in the file. It is likely that you will need a detailed knowledge of the products involved.

The deactivation and uninstallation sections contain lists of instructions of the following types:

- Explicit instructions, for example:

  `Delete the` *`TestWebApp2`* `web application`

- Links to the scripts that were run during deployment, for example:

  `Run the reverse of the` *`myother.mqsc`* `script on the default`
  `queue manager`

  You must perform the reverse actions to those performed by the script, using the reference documentation identified in the instructions.

- Links to manual instructions files used during deployment. You must perform the reverse actions to those contained in the manual instructions files, if when deploying the solution, the manual instructions were actually performed.

Redeploying solution packages

The Solution Deployment Wizard allows you to redeploy all or part of a soluton package that has already been deployed.

You may want to redeploy a solution package because a problem has been found with the package, or perhaps because some elements have changed. Depending on how much of the solution package you need to redeploy, you may need to undeploy the whole package before redeploying. If a complete redeployment is not required, you can use the Solution Deployment Wizard to selectively redeploy elements of the already deployed solution.

Note: Before you select solution elements for redeployment, you may first need to undeploy them. If you generate undeployment instructions for the solution package, you will be able to tell whether undeployment is necessary for individual solution elements.

In the Solution Deployment Wizard, the Package Details panel (see Figure 14 on page 43 shows the deployment status of the elements in the package, allowing you to force the redeployment of elements as required. You must select these elements and select **Update Selected Elements**, then click **Start** to redeploy. The selected elements will be deployed again as though they had never been deployed.

Removing a solution from the Topology Repository

The bizRemoveSolution tool (in the \bin subdirectory where you installed Business Integrator) allows you to remove a solution and its elements from the Topology Repository. This is useful when you are undeploying, or redeploying a solution.

bizRemoveSolution has the following parameters

```
bizRemoveSolution -ID solution name
```

Be careful that you do not remove solution elements that are part of another solution.

Deployment log messages

As solution elements are deployed on their target machines, log messages are generated showing the success or otherwise of the deployment instructions performed during each phase of deployment. The messages are contained in the file AdminMessages.log in the <wsbi install directory>\logs directory.

The messages BIZ2102E and BIZ2103W indicate the number of instructions that:

- failed
- gave unknown status
- gave warnings
- were successful

If deployment is unsuccessful, you should examine the previous deployment log messages on the local and remote machines for the source of the failure, unknown, and warning states. You should then take appropriate action to remedy the problem.

For information about messages generated by the Solution Deployment Wizard, refer to the *WebSphere Business Integrator Messages* book.

Chapter 4. General administration

This chapter provides a description of administrative tools and instructions for performing various tasks for administering and managing the Business Integrator system.

Monitoring and managing the base products

The Product Console Launchpad allows you to launch the administration consoles of base products of Business Integrator from a single machine, allowing you to remotely monitor and manage the components of the base products. Using the Product Console Launchpad you can:

- View all of the machines in the topology, and see which base products are installed on those machines.
- Remotely administer the base products, by launching the appropriate administration console and directing it at the correct machine.
- Automatically log on to the appropriate administration console. Where this is not possible, the Product Console Launchpad provides you with any information needed to log on.
- Register a new administration console for a specified base product.

You can launch the following administration consoles from Product Console Launchpad:

- WebSphere Administrative Console
- DB2 Control Center
- DB2 Client Configuration Assistant
- HTTP Administration Client
- MQSeries Explorer
- MQSeries Services
- SecureWay Directory Management Tool
- LDAP Web Administration

All of the above are stand-alone management consoles, apart from HTTP Administration Client, which has administration tools that run within a Web browser.

An XML document (RegisteredProductConsoles.xml) is used to store data about each product console supported, including information about how the console is launched, and information that is displayed when it is launched. The XML document is pre-configured for each of the base-product consoles that ship with Business Integrator. However, you can use a new console wizard to add new consoles, see "Adding a new console to the launchpad" on page 52.

Using the Product Console Launchpad

To start the Product Console Launchpad click **Start Programs**, then **IBM WebSphere Business Integrator**, then **Product Console Launchpad**.

The left-hand pane of the Product Console Launchpad shows a tree view of the topology, and you can select a logical, physical, or solution view. The topology tree allows you to navigate the topology and display the node that you want to monitor and manage. A node corresponds to a machine, facility, or base product (physical view), a facility or base product (logical view), or a solution element or artifact (solution view).

Figure 15 shows the Product Console Launchpad with a physical view of the topology tree.

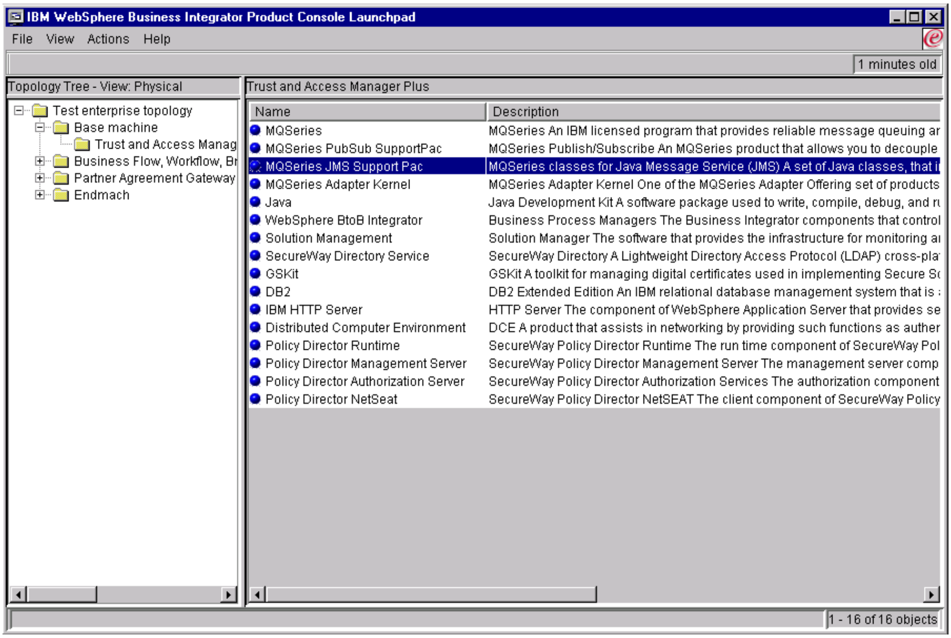


Figure 15. Product Console Launchpad

The right-hand pane of the Product Console Launchpad shows the details of the node that is currently selected in the topology view.

The action bar on the Product Console Launchpad contains the **File**, **View** and **Actions** menus.

From the **File** menu, you click **Close** to exit from the Product Console Launchpad.

From the **View** menu, you select logical, physical, or solution views of the topology as required. You click **Refresh Topology** to update the topology information.

From the **Actions** menu you click:

Add New Console
To launch the new console wizard, see "Adding a new console to the launchpad" on page 52.

console name
To launch the administrative console for a product. The consoles you can launch for a product are only listed on the **Actions** menu when you select the product in the topology tree or right-hand pane.

The counter at the top right of the console shows the time since the information for the selected node was retrieved from the Topology Repository. The counter is reset when you select **Refresh Topology**. However, the counter is also reset for the selected node when you change the type of view (logical, physical, solution).

To monitor and manage products, you can also select the node in the topology tree, or the details pane, and right-click to display the options available in the **Actions** menu.

To display the properties for a selected node, right-click and select **Properties**, see Figure 16 on page 52. These are the properties stored in the Topology Repository for the selected node.

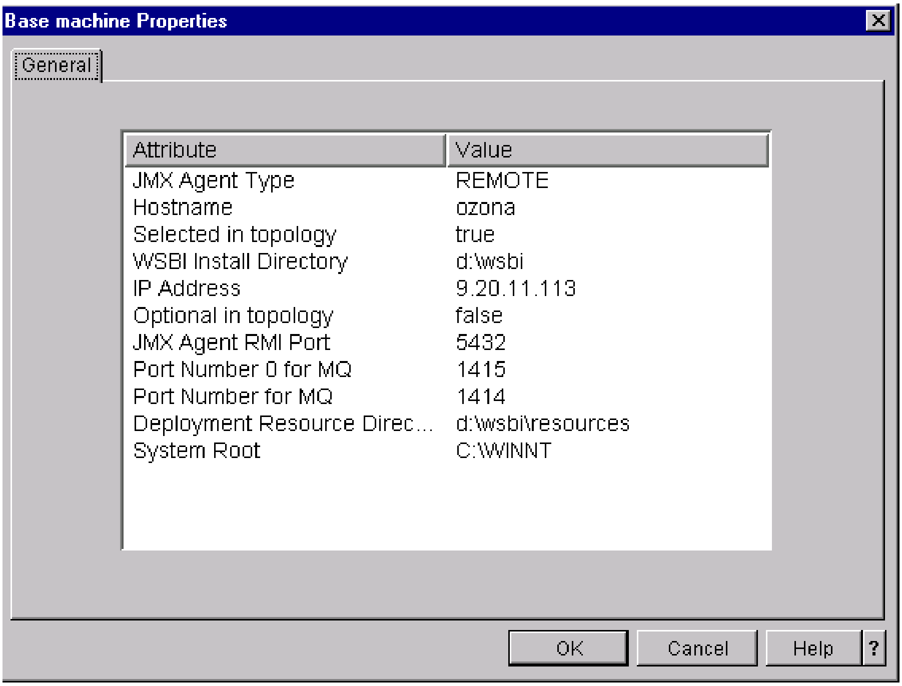


Figure 16. Properties panel

For more information about using the Product Console Launchpad, refer to the online help.

Adding a new console to the launchpad

The Product Console Launchpad contains a new console wizard that allows you to add a new product console to the Product Console Launchpad. You may want to do this to use a new console for an existing product, or to add a console for a new product that has just been installed on a remote machine in the topology.

The new console wizard prompts you for information about the console, (for example, the executable to be run when users launch the console, and any command-line arguments that are needed), and automatically updates the XML document. When you add a new console, you can start it via the Product Console Launchpad immediately—you do not have to restart the Product Console Launchpad for the change to take effect.

For information about using the new console wizard, refer to the online help.

Editing and deleting console definitions

Business Integrator does not provide a tool that allows you to edit a console definition, or delete a console definition.

To delete a console that has previously been registered, you must use a text editor to delete the appropriate section in the RegisteredProductConsoles.xml file.

To change a console definition, you can either:

- Edit the RegisteredProductConsoles.xml file directly, or,
- Delete the appropriate section from the XML file document and re-register the console using the new console wizard.

For information about the console.dtd, refer to "The console.dtd file".

The console.dtd file

The registered consoles file, RegisteredProductConsoles.xml, contains definitions for the consoles registered for use with the Product Console Launchpad. The format of this file is defined by the console.dtd shown below:

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- DTD for product consoles -->

<!ELEMENT topologytypes (topologytype*)>
<!ELEMENT topologytype (command+)>
<!ATTLIST topologytype
  type CDATA #REQUIRED>
<!ELEMENT command (path, menuoptionstring, cla*, errorinfo?, userinfo?, resourcename?)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT menuoptionstring (#PCDATA)>
<!ELEMENT cla EMPTY>
<!ATTLIST cla
  arg CDATA #IMPLIED
  value CDATA #IMPLIED>
<!ELEMENT errorinfo (#PCDATA)>
<!ATTLIST errorinfo
  isTag (yes|no) #REQUIRED>
<!ELEMENT userinfo (#PCDATA)>
<!ATTLIST userinfo
  isTag (yes|no) #REQUIRED>
<!ELEMENT resourcename (#PCDATA)>
```

The XML file is read when the Product Console Launchpad initializes, and associates consoles with specific applications.

You can register new consoles with the new console wizard, which automatically updates the XML file, see "Adding a new console to the launchpad" on page 52.

console.dtd elements

The following sections describe the elements and attributes in console.dtd.

<topologytypes>: This element is a container for a number of <topologytype> elements.

<topologytype>: This element describes the consoles available for administering a product. It contains the child element <command>, and the attribute type. The type attribute describes the type of the product as it is stored in the topology.

For example:

```
 <topologytypes>
   <topologytype type="MQ_QMGR">
     <command>
     ...
     ...
     </command>
   </product>
```

<command>: This element contains the command that is used to launch the console. Each product may have at least one command associated with it; there is at least one command for every console available to administer the product. <command> contains the child elements <path>, <menuoptionstring>, <cla/>, <errorinfo>, <userinfo>, and <resourcename>.

For example:

```
<command>
  <path>C:\Winnt\System32\mmc</path>
  <menuoptionstring> - QueueManager </menuoptionstring>
  <cla arg="/s"
  value="C:\Program Files\IBM WebSphere Application Integrator\MQSeries\MQSeries.msc"/>
  <resourcename>
   com.ibm.btobi.managagement.launchpad.RegisteredProductConsoles
  </resourcename>
</command>
```

<path>: This element describes the path associated with the command. This is the fully-qualified path and filename that is used to run the required console application. For example:

```
<path>C:\Winnt\System32\mmc</path>
```

<menuoptionstring>: This element contains the string for the menu option that the user selects to run the required console. For example:

```
<menuoptionstring>MQSeries Explorer - QueueManager</menuoptionstring>
```

<cla>: This element defines command line arguments and their values to be passed to the console when it is launched. For example:

```
<cla arg="/s" value="C:\Program Files\IBM WebSphere Application Integrator\MQSeries.msc"/
```

<cla> is an empty element that is completely described by its attributes, arg and value.

The arg attribute specifies the command line argument.

The value attribute specifies a value for the argument, and can identify values from the Topology Repository, for example:

```
<cla arg="-h" value="@{top:type=ComputerSystem,name=hostname}"/>
```

which would specify the remote machine's hostname.

<errorinfo>: This element contains information that is displayed to the user if there is an error condition when the console is launched. The isTag attribute specifies whether or not the information is obtained form a resource file. In this case, <errorinfo> specifies the name tag of a string in a resource file specified by the <resourcename> element. For example:

```
  <errorinfo isTag="yes">MQSeriesExplorerErrorInfo</errorinfo>
```

<userinfo>: This element contains information that must be entered by the user when the console is launched. This information allows the user to administer the product on the appropriate machine. The isTag attribute specifies whether or not the information is obtained form a resource file. In this case, <userinfo> specifies the name tag of a string in a resource file specified by the <resourcename> element. For example:

```
  <userinfo isTag="yes">MQSeriesExplorerUserInfo</userinfo>
```

In this example the string `MQSeriesExplorerUserInfo` would cause the following to be displayed to the user:

```
You wish to look at the queue manager called queue_manager_name
If it does not appear in the list in the left hand pane, you will need to
add it by right clicking on the Queue Managers object in the tree and
selecting 'Show Queue Manager'.
Then select 'Show a Remote Queue Manager' and enter queue_manager_name
as the queue manager name and hostname as the connection name.
Click the 'OK' button to add the new queue manager to the list.
```

<resourcename>: This element defines the name of a resource file that contains user information and error information strings. For example:

```
<resourcename>
  com.ibm.btobi.managagement.launchpad.RegisteredProductConsoles
</resourcename>
```

The resource file contains strings pointed to by tags in the <userinfo> and <errorinfo> elements.

Administration consoles of other products

The console of some base products are not included in the Product Console Launchpad, but are optionally installed on the same machine as the Product Console Launchpad These are:

- Message Broker Console.

  This is the Control Center for MQSeries Integrator that is installed with the Message Broker facility, which is optional in some topologies and required in others.
- PAM Console,

  The console for Partner Agreement Manager, which you can use to:
  - Add new trading partners
  - Suspend existing trading partners
  - Reassign trading partners to partner groups and public processes
  - Start, stop, and query the state of processes, adapters, and channels
- EDI Console.

  The console for DataInterchange which is an option in some topologies.

Refer to the documentation for MQSeries Integrator, Partner Agreement Manager, and DataInterchange for information on using the consoles, see "Bibliography" on page 191.

Monitoring and managing the Business Integrator platform

The Platform Console allows you to manage resources and applications in the Business Integrator system from a single point.

Using the Platform Console you can:

- Start and stop base products, such as WebSphere Application Server, and view their status
- View the status of machines in the topology
- View the status of facilities in the topology
- View the status of solutions and their elements

The management actions that you can take, depend on the MBeans that are provided for the resources.

Using the Platform Console

To start the Platform Console click **Start Programs**, then **IBM WebSphere Business Integrator**, then **Platform Console**.

The left-hand pane of the Platform Console shows a tree view of the topology, and you can select a logical, physical, or solution view. The topology tree allows you to navigate the topology and display the node that you want to monitor and manage. A node corresponds to a machine, facility, or base product (physical view), a facility or base product (logical view), or a solution element or artifact (solution view).

Figure 17 shows the Platform Console with a logical view of the topology tree.

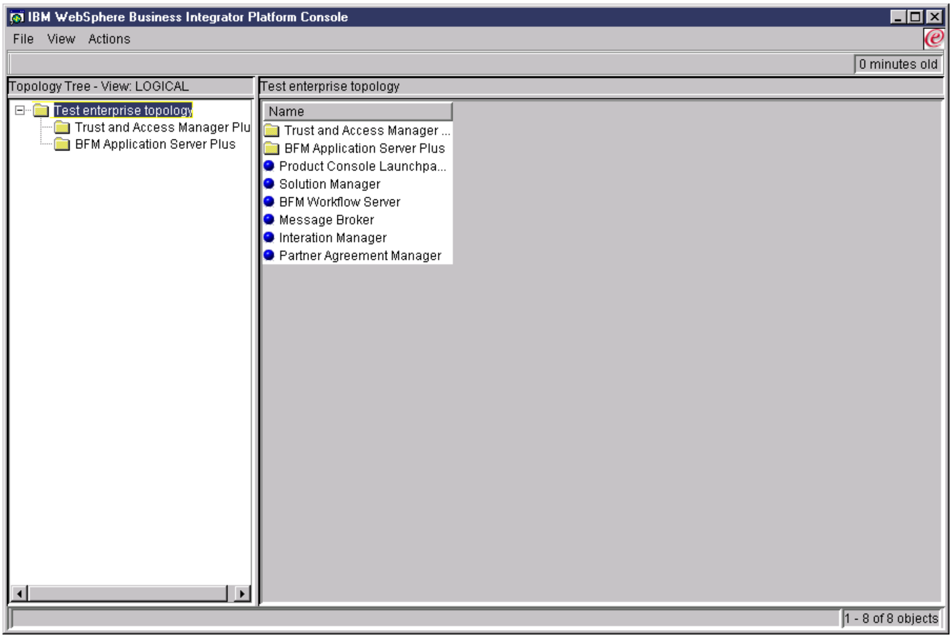


Figure 17. Platform Console

The right-hand pane of the Platform Console shows the details of the node that is currently selected in the topology view.

The action bar on the Platform Console contains the **File**, **View** and **Actions** menus.

From the **File** menu, you click **Deploy Solution** to launch the Solution Deployment Wizard, see “Using the Solution Deployment Wizard to deploy the solution package” on page 42, and **Close** to exit from the console.

From the **View** menu, you select logical, physical, or solution views of the topology as required. You click **Refresh Topology** to update the topology information.

From the **Actions** menu you click **Manage**, to start a dialog to manage or monitor the currently selected node. You cannot monitor and manage every node in the topology tree, as not every node is manageable.

The counter at the top right of the console shows the time since the information for the selected node was retrieved from the Topology Repository. The counter is reset when you select **Refresh Topology**. However, the counter is also reset for the selected node when you change the type of view (logical, physical, solution).

To monitor and manage nodes, you can also select the node in the topology tree, or the details pane, and right-click to display the **Manage**, and **Properties** options.

In the Management Properties panel, (see Figure 18), you can view and edit the properties of the node, and perform the operations that are possible for the node.

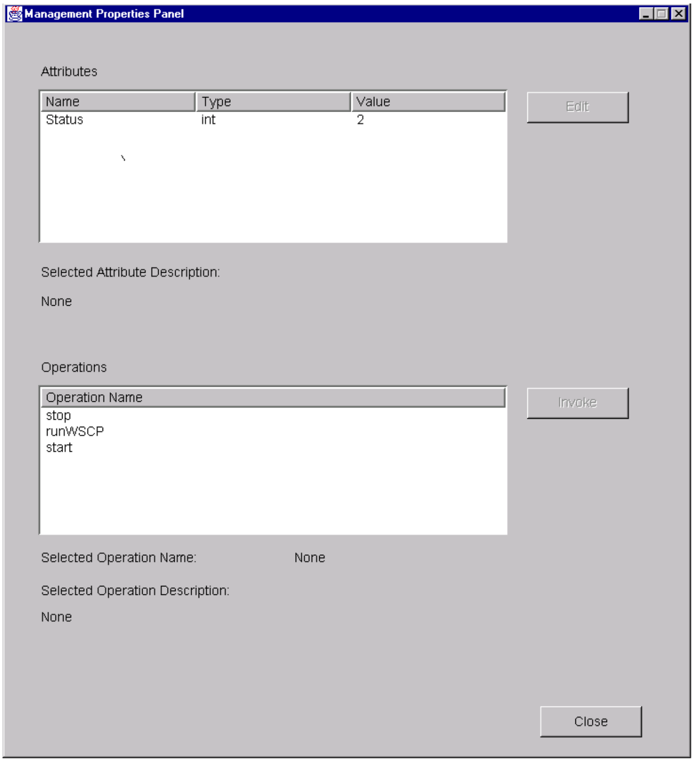


Figure 18. Management Properties panel

For more information about using the Platform Console, refer to the online help.

Components that you can monitor and manage

Using the Platform Console you can:

- Start and stop Windows NT services, such as MQSeries services and DB2 services. For more information about services that are installed with Business Integrator, see "Starting and stopping services".
- Start and stop WebSphere applications.

Starting and stopping services

Many services and Business Integrator managers are configured to start automatically when the system is started. Some of the services and managers start after WebSphere Application Server starts.

A few services must be manually started each time the entire system is stopped and started. Review Table 1 to see how a particular service or manager is started. Some services and managers require that other services or application servers must already be started. This table assumes the entire system is being started.

Table 1. Services and Business Integrator managers startup

Manager or Service	Automatic Start with a System Boot	Manual Start Required	Prerequisite Servers or Services
DataInterchange Services	No	Yes. See "Starting DataInterchange Services" on page 60.	
DB2	Yes	No	
LDAP	Yes	No	
Interaction Manager	No	Yes. Start from the WebSphere Administrative Console.	Trust and Access Manager
MQSeries Workflow 3.3 FMC NT service	Yes	No	
MQ Workflow Java Agent	No	Yes. Start the MQWF Java Agent using theWebSphere Administrative Console	

Table 1. Services and Business Integrator managers startup (continued)

Manager or Service	Automatic Start with a System Boot	Manual Start Required	Prerequisite Servers or Services
Partner Agreement Manager Services (PAMAS NT service)	Yes	No. Can be started using the WebSphere Administrative Console. Appears as `pamAppServer` on the console.	
Policy Director Authorization NT Service	Yes	No	
Policy Director Management NT Service	Yes	No	
Solution Manager Services	No	Yes. See “Starting Solution Manager services” on page 61.	
Trust and Access Manager	No	Yes. Start from the WebSphere Administrative Console.	Solution Manager Server.
WebSEAL	Yes	No	
WebSphere AdminServer NT service	No	Yes. Start from the Windows NT Services panel.	
WWFServices	Yes	No	

Starting DataInterchange Services

1. Verify that DB2 and MQSeries Windows NT services are running. These services should always be running on the WebSphere Application Server.
2. Run the followng batch command file to start the DataInterchange Daemon:

   <DI_install_path>\bin\DIAdapterDaemon.bat. This starts the MQSeries Adapter Kernel adapter daemon for the application ID DIDaemon.

Stopping DataInterchange Services

To stop the DataInterchange services access the MDOS Window `DIDaemonServer`; then use Ctrl Break.

Starting Solution Manager services

1. Verify that DB2 and MQ Series Windows NT services are running. These Windows NT services should always be running on the WebSphere Application Server.
2. From the WebSphere Application Server Administration Console, start the Solution Manager server.
3. Start the Java Message Service by issuing the following commands:

   ```
   strmqbrk -m<Queue_manager_name>
   runmqlsr -t tcp -p<port_number>
   ```
4. Run the following batch file <install_path>\utils\setSMenv.bat
5. Run the following batch files to first register and then start the Audit Log server and the Exception Management server:
   a. <install_path>\utils\registerAuditLogServer.bat
   b. <install_path>\utils\registerExceptionServer.bat
   c. <install_path>\utils\StartAuditLogServer.bat
   d. <install_path>\utils\StartExceptionServer.bat

Stopping Solution Manager services

To stop a specific Solution Manager Service use one of the following:

- To stop the Exception Management server run these batch files:
  1. <install_path>\utils\setSMenv.bat
  2. <install_path>\utils\StopExceptionServer.bat
- To stop the Audit Log server run these batch files:
  1. <install_path>\utils\setSMenv.bat
  2. <install_path>\utils\StopAuditLogServer.bat
- To stop the Java Message Service access the MSDOS Window `runmqlsr`; then use Ctrl Break.

Accessing the Solution Manager Console

For Business Integrator Enterprise configuration:

1. Type **https://WebSealhostname** in your browser and press Enter.
2. Logon as an Adminstrator. (Default ID is WSBIAdmin) or as a Manager.
3. Select the Exception Console or Trace Console menu item from the Navigation pane on your browser.

For Business Integrator Entry configuration:

1. Type **http://hostname/ePortal/servlet/Portal?Action=Logon&Solution=BtoBiTemplates** in your browser and press Enter.
2. Logon as an Adminstrator. (Default ID is WSBIAdmin) or as a Manager.

3. Select the Exception Console or Trace Console menu item from the Navigation pane on your browser.

Using Solution Manager XML extensions

At times a solution may need to handle very large XML messages or documents. A typical scenario might be:

1. A purchase order is submitted to a supplier to provide a large number of different parts. The order has many different quantities, part numbers, part descriptions, and prices.
2. The supplier filling the order returns an advanced shipping notice. This notice might include: Purchase orders filled in the shipment, which parts are shipped, which parts are on back order, new prices for the parts, substitute parts and other data as appropriate to the purchase order.
3. Due to the large size and the number of fields in the XML document, parsing specific details of the advance shipment notice is a long process and can affect system performance. XML extensions for DB2 enables the user to efficiently search for specific details and perform complex queries on the content of the XML document. To accommodate these large documents LDAP configuration for Solution Manager is required and DB2 XML extensions have to be installed.

These extensions are provided in DB2 version 7. Solution Manager uses JDBC2 driver fixpacks for DB2. XML extensions and access beans must be installed:

Use the following procedure to install fixpack3:

1. Backup the existing epicalog database using the database control center. Instructions on how to back up your database is provided in DB2 books. See "Other Libraries" on page 193. Select Backup, then database and chose a directory.
2. Logon as the DB2 administrator, (userid = db2admin, password = db2admin.
3. Install fixpack3 on the drive and computer where DB2 is installed.
4. Edit the batch file `xmlcreatesample.bat`. This file is located in <install_path>\sql. Verify the paths and the drive settings in the batch file match the drive letter and paths where DB2 is installed.
5. Create a dtd file and a dad file, then copy these files to a directory on the install drive. Information about creating a dad file is provided in DB2 books for XML extensions and a sample of an Advance Shipment Notice is provided in the ASN.dtd and ASN.dad files. These files are located in <install_path>\xml

6. Open a DB2 command window and run the edited batch file `xmlcreatesample.bat`. Verify in DB2 in epicalog that the tables the batch file should create were added to the epiclog database.
7. Use the Directory Management tool to make the following change in LDAP.
   a. For the applicationID **epicLogServer**, in the ePICSolMgrGroup=XMLColumns, add an ePICSolMgrGroupMember with the name of your specific message's *BodyCategory*. The provided sample uses MDXML as the name.
   b. Add an ePICSolMgrGroupSubMember with the name of your message's *BodyType*. The provided sample uses ASM as the name.
   c. In the *valid value* field enter the name of your XML column. The provided sample uses *ASNMD* for the name.
8. Add an XML message with your defined BodyCategory and Bodytype.
9. Run the batch file `StartAuditLogServer.bat`. Verify that the rows were added to your XML defined column in the database.

Archiving for Audit Logging

If the Audit Logging database grows too large, searches can become very slow. To prevent the database from becoming too large, a local archiving policy should exist to handle moving data from the database to other unsearchable media for storage. The database administrator should set up the Business Integrator Audit Logging database as read and write only without, delete or update capabilities. Creating archives and movement of archived logs to a database must be performed by the customer using their existing database tools. For the Business Integrator example that follows, the supported NT operating system with a DB2 database for the run time environment are assumed.

As installed, three tables exist in the Audit Log (ePICLOG) database:

ePIC2
: This is the current table used by the Audit Log server and is searchable. When this table is full it is moved to the ePIC2ARC table.

ePIC2ARC
: This is a temporary table that has the next set of data to be archived.

Temp
: This table provides a place for archived data that may need to be retrieved and searched. See "Procedures to unarchive data for searching" on page 64

Example procedures for archiving

To archive logged data, do the following:

1. Pause the database server by issuing the pause command from a DOS prompt.
2. Using the DB2 export command, do the following:
   a. Export the data from the ePIC2 populated table to a file named epic2arc.ixf
   b. Export the data from the ePIC2ARC table to a file named *today's date*.ixf, where *today's date* is the date when the archiving takes place.
3. Delete the contents of the tables using the database delete command.
4. Import the file epic2arc.ixf into the ePIC2ARC table.
5. Start the Audit Logging database using the restart command.

Procedures to unarchive data for searching

Archived data is not searchable. To make the archived data available for searching, you must reload the archived data from the archive file into a second temporary table in the database. To unarchive the data, do the following:

1. Pause the database server by issuing the pause command from a DOS prompt.
2. Using the DB2 import command, import the .ixf files (for example, *01022001*.ixf, where *01022001* is the date when the data was archived) containing data you want to search into the Temp table in the Audit Log database.

When you are finished with the data, delete the contents of the Temp table in the Audit Logging database.

Registering users

Users might have to be registered in these places:

- DataInterchange. See the *WebSphere Business Integrator DataInterchange for Windows NT User's Guide*.
- Partner Agreement Manager. Use the PAM Process Manager. For details, see the "WebSphere Partner Agreement Manager library" on page 193.
- LDAP for access to the trusted zone of Business Integrator. See "Appendix A. Business Integrator LDAP" on page 77.
- Solution. See the sample solutions at URL: http://www-4.ibm.com/software/webservers/btobintegrator/.
- Each endpoint application, depending on the requirements of the specific endpoint application.

Configuring topology properties

If you need to change topology properties, you can edit the tmapi.properties file, which is in the \settings subdirectory where you installed Business Integrator. You might want to edit this file, for example, if the URL for the machine containing the Topology Server changes, or if you need to add or change a username and password for accessing the Topology Server.

The properties are:

webDav.url
: Specifies the URL and alias for the Topology Server. Note that you should not include `http://` at the beginning of the URL. For example: webDav.url=integrator/top

webDav.username
: Specifies the username for accessing the Topology Server.

webDav.password
: Specifies the password for accessing the Topology Server.

webDav.repository
: Specifies the path of a directory on the local machine that is used to store a local copy of the topology.xmi file. For example:

```
webDav.repository=C://Program Files//IBM//WebSphere Business Integrator//topology//local
```

webDav.predefined
: Specifies the path to a directory to contain the predefined topologies. For example:

```
topology.predefined.path=..//..//topology//predefined
```

Setting up SSL and using digital certificates

To use Secure Sockets Layer (SSL) security, you must obtain a digital certificate signed by a trusted Certificate Authority (CA), or create a self-signed certificate.

Using a certificate from a trusted CA has the advantage that most SSL communication partners know and trust that CA, and will, therefore, most likely (depending on their configuration) accept a new certificate. This is especially helpful when communicating with partners outside your organization and beyond your authority to change security options. However, it can take several days, or up to two weeks to obtain a certificate from a trusted CA.

If you do not have a certificate from a trusted CA at the time you install Business Integrator, you can use a self-signed certificate until you obtain the certificate from the CA. This is described in the *WebSphere Business Integrator Installation Guide*. The disadvantage of using self-signed certificates is that each client or server using this kind of certificate needs to have the new root certificate imported, which may create an administrative burden.

Using a certificate signed by a trusted CA

Using a certificate signed by a trusted CA involves the creation of a key database and a certificate request that is then sent to the CA. After the certificate is received from the CA, it must be stored in the key database. The following describes how to perform these steps using the gsk4ikm utility.

1. Create the server key database (.kdb file):
   a. Start the gsk4ikm utility
   b. On the main window, click **Key Database File**, then click **New**.
   c. Click **CMS key database file** from the **Key database type** list, and then type in the name and location of the key database file to be created. This file has an extension of .kdb, for example, `ldap_key.kdb`. Click **OK** to close the dialog.
   d. In the next dialog box, enter a password, an expiration time if required, and make sure you select the **Stash the password to a file?** check box, otherwise, SecureWay Policy Director cannot open the database file. Click **OK** to close the dialog. The password is then encrypted and stored in a file with the same name as the key database file but with an extension of .sth. Your database file is now created.
2. Create a certificate request:
   a. On the main window, click **Create**, then click **New Certificate Request...**. In the dialog box, enter the following:
      - In **Key label**, a clear, descriptive label for the certificate)

- In **Key size**, 512 or 1024, depending on security requirements and the language version of the gsk4ikm utility.
- A **Common name**
- **Organization** and other pertinent information to identify the owner of the certificate
- The full path name for the certificate request file

b. Click **OK** to create the request.

c. The file created (default file name certreq.arm) contains the certificate request.

3. Send the certificate request, that is, the certreq.arm file, to the certificate authority of your choice by mail or from their Web site (following their instructions).

   You may now have to wait up to two weeks for the CA to process and return your certificate. In the meantime, you can enable SSL security by creating, storing, and importing a self-signed certificate. This is described in the *WebSphere Business Integrator Installation Guide*.

4. When you have received the certificate from the CA, store it in the key database file:

   a. Start the gsk4ikm utility.

   b. On the main menu make sure that your key database file is open (check the filename in the Key database information portion of the window). If it is not open, click **Key Database File**, then click **New**. and open your file.

   c. Select **Personal Certificates** from the selection list in the lower Key database content portion of the window.

   d. Click **Receive...** on the right of the window.

   e. Enter the information about the file containing the signed certificate and click **OK**. This adds the certificate to the key database file. You will see the new certificate in the list under **Personal Certificates**.

5. If required, store a root certificate of the CA in the key database file.

   By default, root certificates of the most common CAs are already present in the file; so, you do not need to add them again. A trusted root is simply an X.509 certificate that has been signed by a trusted entity (for example, Verisign). You can see what root certificates there are by selecting **Signer Certificates** from the selection list in the Key database content portion of the main window. If your CA is not present in that list, you must obtain a root certificate from this CA and add it by clicking **Add...** on the right of the window.

Chapter 5. Problem determination

You can diagnose problems and retrieve debugging information from the log databases generated by the logging services. The logging service is used in the Business Integrator system to trace and log errors and events that occur throughout the system.

Business Integrator differentiates between three types of log messages:

Audit The audit log records messages that are sent between applications and adapters. These messages are generated during normal operation.

Trace Trace messages are generated during troubleshooting and affects performance.

Exceptions
Exceptions are sent to the Exception Manager for handling. For example, if an application stops abnormally, the Solution Manager can restart without manual intervention. These messages are generated during normal operation.

The name and location of the log files are defined in the Directory. The log file for tracing is defined by the TRACE_MESSAGE_FILE attribute. The log file for audit logging is defined by the LOG_MESSAGE FILE attribute.

Solution Exceptions

Solution exceptions are handled by an exception Daemon or the Java Message Service Listener, a component of Business Flow Manager. The exception server publishes user-defined commands to JMS Listener. The exception manager stores system exceptions in the exception manager database.

Solution exceptions are published in a user-written XML file (`exception.dtd`). A typical solution exception might be, a user submits a purchase order and does not receive a response within a predefined time. The reason for the lack of response might be that the person authorized to approve the purchase order has not yet marked it as approved.

The Exception manager filters exceptions by examining exception type values stored in LDAP. LDAP looks for the exception type in the Application ID field, routes the exception to the appropriate application, and then notifies the exception manager that the state of the exception is changed to resolving. Exception Manger routes the exception to JMS listener. The involved application is responsible for changing the exception to the open or closed state. If LDAP cannot identify or locate the application to handle the exception, LDAP reports an open problem to exception manager.

An *update* field in LDAP controls whether a user can delete errors from the log. The default setting of this field is *delete*.

Solution exceptions are viewed by administrator logon to the Solution console and then selecting the Exception console with a browser through Interaction Manager.

There is a Data Access Object utility that can be used to access logs from a solution. For information on how to use this utility, see "Related documentation" on page 192.

Audit logging

The Audit Log service records the flow of the messages sent between applications through the Business Integrator infrastructure, providing a centralized logging facility. These records are stored in the log database in the order they are received.

The Audit log contains the following information:

Unique time stamp
: A key that uniquely identifies the message in the log.

Message unique ID
: The unique identifier of the message.

Message Destination ID
: The name of the application (for example, SAP or i2) that is to receive the message.

Message Source ID
: The name of the application (for example, SAP or i2) that generated the message.

Message Transaction ID
: The unique identifier of the transaction created when the message was received. This field is optional.

Message time stamp
: The date and time the message was generated.

Message type
: The type of message to be logged.

Log source ID
: The identifier of the application that logged the message.

Log time stamp
The date and time the message was logged.

Body category
The category of the message (for example, RosettaNet).

Body type
This parameter is used by the target application to identify the parsing mechanisms needed to recover the original information.

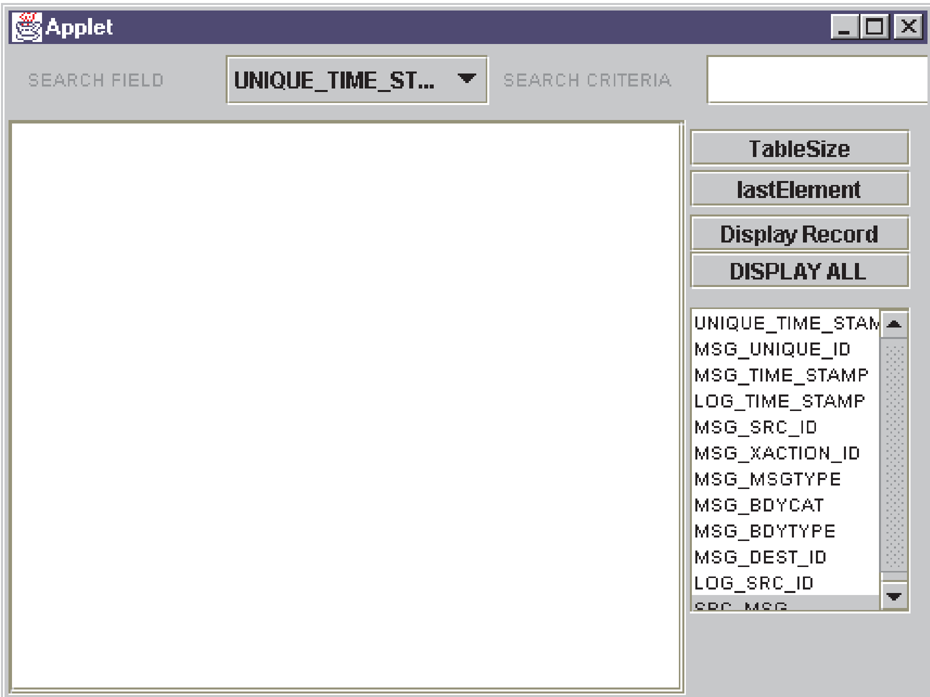
Source message
The description of the message.

Audit Log server operation

Viewing the Audit Log

You can view the audit-log database using the Solution Console (see the following figure). The Solution Console allows you to view the number of records in the database, view all records, and view a subset of records based on a simple search.

Figure 19. Solution Console.



Starting and stopping the Solution Console

See "Starting and stopping services" on page 59 for information on starting and stopping the Solution Console. See "Accessing the Solution Manager Console" on page 61 for details about adminstrator logon to the Solution Console.

Displaying the number of records

To display the number of log records in the audit-log database, click on **Table Size**. The number appears in the workspace on the left.

Displaying all records

To display all the log records in the audit log, select the fields you wish to display from the scroll list and click on **Display All**. This list of log records appears in the workspace on the left.

Displaying a subset of records

The Solution Console allows you to display a subset of log records based on a search. Currently, you can search on only a single field.

To display a subset of records, perform the following steps:

1. Select the fields you wish to display from the scroll list
2. Select the field to search on from the **Search Field** drop-down list.
3. Type the text to search for in the **Search Criteria** field.
4. Click on **Display Record**. The log record appears in the workspace on the left.

Displaying the last record

To display the last record that was added to the log, select the fields you wish to display from the scroll list and click on **Last Record**. The log record appears in the workspace on the left.

Tracing

You can diagnose problems and retrieve debugging information from trace logs generated by clients of the Trace service. The Trace service is used on the Business Integrator platform to trace and log events that occur on the system. It is a network-logger daemon that consists of a Trace server and clients. The Trace server collects log messages from each Business Integrator manager (client) and stores them in a persistent repository or displays them to the console. These log files are used to diagnose and resolve problems with any manager in the Business Integrator system.

The Trace server writes the log records to:

- The console in a command prompt window
- A flat file

- A circular file set
- A socket
- An MQSeries queue, that allows you to forward the log records to more than one location mentioned in this list.

For additional information on Trace logs see, *IBM MQSeries® Adapter Kernel for Multiplatforms Problem Determination Guide*, GC34-5897.

Viewing the trace log

You can view the trace log from a flat file or a circular file set using an ASCII editor.

Clearing the trace log

The trace log is not automatically purged. When the log become too large, the file must be backed up and then cleared.

To clear a log file, other than a circular file, type **echo "" >log_filename**, where *log_filename* is the name of the trace-log file, or simply delete the file. If you delete the file, the Trace server creates a new one.

Exception logging

Exceptions are errors or events that occur unexpectedly or infrequently and are sent, in the form of messages, to the Exception service for handling. The Exception log is the repository for exception messages and is accessible through the Solution console. The Exception log contains the following information:

ExceptionID
The unique identifier of the exception.

ExceptionTypeID
The unique identifier of the exception type.

Description
A description of the exception.

Priority
The priority of the exception.

Status The status of the exception (for example, open or closed).

TimeOpen
The time the exception was logged.

TimeClose
The time the exception was closed.

ExceptionSource
The source of the exception.

ExceptionHandler
The handler class for the exception.

Exception server operation

Viewing the Exception log

You can view the exception log from the Solution console or using SQL query commands from the DB2 command prompt.

Note: You must have administrative privileges to view the exception log.

Clearing the Exception log

After you resolve a problem associated with an exception, you must manually remove the exception from the Exception console by performing the following steps:

1. Select the exception that was resolved.
2. Press the **Del** key.

Analysing topology problems

Business Integrator provides a utility program, bizTmapiUtility, for analysing topology problems. You should use this utility under the direction of IBM service personnel. The utility:

- Performs diagnostic tests to check that the Topology Server can be accessed
- Lists the contents of the Topology Repository in a more easily read form than the topology.xmi file
- Unlocks a locked topology.xmi file

For more information about topology, refer to the *WebSphere Business Integrator Concepts and Planning* book.

Running the utility

You run the utility from the \bin subdirectory of the directory where you installed Business Integrator. The parameters are:

```
bizTmapiUtility [-diagnose|-unlock|-lock|-list]
                [-username username]
                [-password password]
                [-url url]
                [-repository directory]
```

where:

-diagnose
Runs diagnostic tests that determine whether the Topology Server can be accessed.

-unlock
Unlocks the topology.xmi file in the Topology Server if it is locked.

-lock
Locks the topology.xmi file in the Topology Server.

-list
Displays a list of all the objects in the topology, including topology types and topology objects.

-username *username*
Specifies that *username* is used instead of the value from the tmapi.properties file. This is the username for accessing the Topology Server.

-password *password*
Specifies that *password* is used instead of the value from the tmapi.properties file. This is the password for accessing the Topology Server.

-url *url*
Specifies that *url* is used instead of the value from the tmapi.properties file. This is the URL and alias for accessing the Topology Server, and should not include `http://` at the beginning.

-repository *directory*
: Specifies that *directory* is used instead of the value from the tmapi.properties file. This is a directory on the local machine that is used to store the retrieved topology.xmi file.

The properties set in the tmapi.properties file are used with the tool unless the value is overidden by the `-username`, `-password`, `-url`, or `-repository` parameters. For information about the tmapi.properties file refer to "Configuring topology properties" on page 65.

Appendix A. Business Integrator LDAP

This section describes the Business Integrator schemas and directory trees. The schema defines the content of the User Registry and the registry contains information on users that require access to applications. Users are defined as people, applications, and Business Integrator managers, such as Interaction Manager and Business Flow Manager.

Business Integrator Directory

The root suffix is **o = ePIC**. Entries under this node are groups that contain Business Integrator applications, solutions, awarelets, and administrative users. Figure 20 on page 78 shows a compressed view of the directory.

CAUTION:
Do not change the LDAP entries described here. This information is provided only for reference. Changing LDAP entries or their attributes can cause system failures. A small number of attributes can be changed. Those attributes, how to change them and why they might be changed will be described in the context of that specific part of the directory.

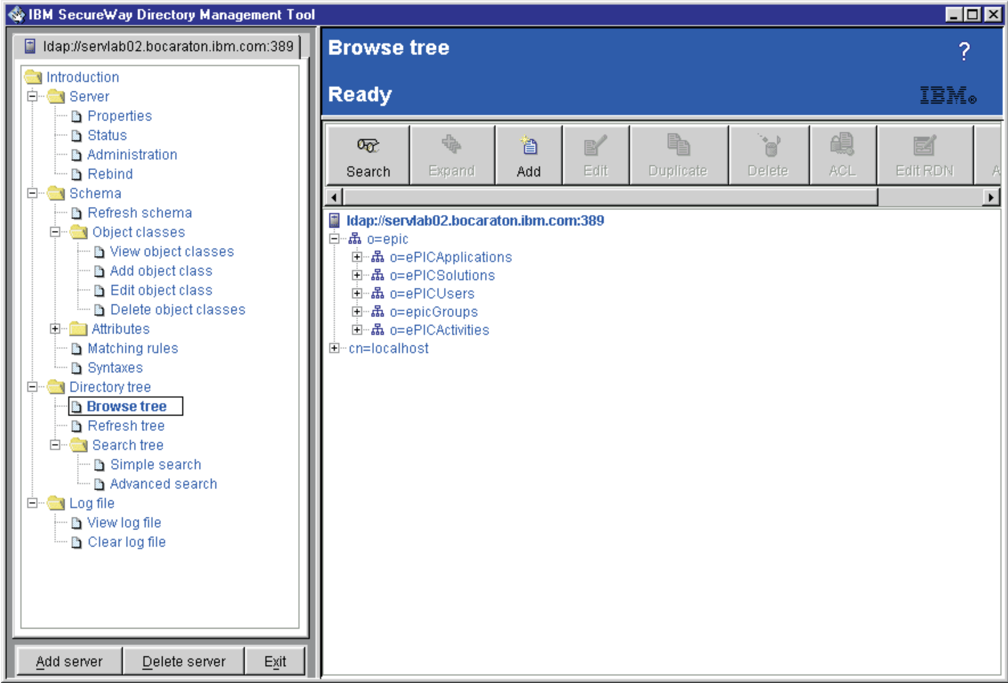


Figure 20. Business Integrator Directory Tree (Compressed View)

The following entries off the epic root are described in:
- "Applications Tree (ePICApplications)"
- "Solutions Tree (ePICSolutions)" on page 133

Applications Tree (ePICApplications)

The object class for Business Integrator applications is **ePICApplications** This object class provides the following common elements:

ePICAppId

The name of the application.

ePICLogging

Defines whether audit logging is performed.

ePICTrace

Defines whether tracing is used.

ePICTraceLevel

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897.

ePICTraceClientid

Defines the name of the trace client configuration that is used. The default is *TraceClient*.

ePICExceptionErrorLog

Defines whether exception logging is performed.

ePICExceptionWarningLog

Defines whether warning errors are logged.

ePICExceptionInfoLog

Defines whether exception logging for informational messages is performed.

ePICHostname

The name of the server that hosts this application.

ePICPropertyFile

A file that contains configuration parameters for this application.

ePICLogonInfoClassName

Defines the name of the logon class that is used to connect to the application when using an EAB adapter.

Figure 21 on page 80 shows the ePICApplications tree.

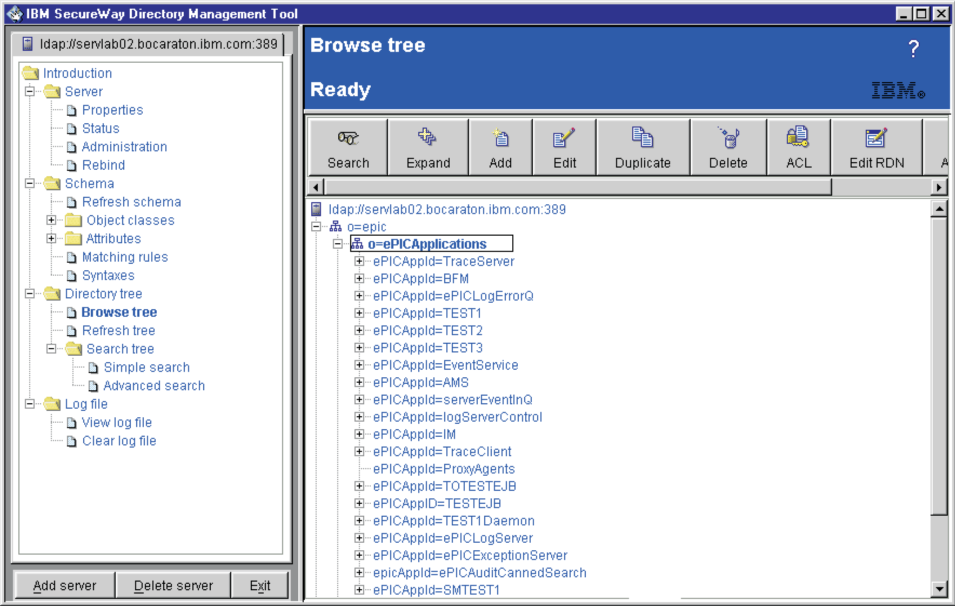


Figure 21. ePICApplications Tree Entries

The entries in the ePICApplications tree are described in the following sections:

- “TraceServer” on page 81
- “Business Flow Manager” on page 84
- “LogErrorQ” on page 86
- “EventService” on page 89
- “Trust and Access Manager” on page 91
- “serverEventInQ” on page 96
- “logServerControl” on page 99
- “Interaction Manager” on page 102
- “TraceClient” on page 104
- “ePICLogServer” on page 108
- “Exception Server” on page 112
- “ePICAuditCannedSearch” on page 127

TraceServer

TraceServer has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=TraceServer
      cn=ePICAppExtensions
          ePICTraceHandler=com.ibm.logging.SocketHandler
          ePICTraceHandler=com.ibm.logging.ConsoleHandler
          ePICTraceHandler=com.ibm.logging.MultiFileHandler
      cn=ePICAdapterRouting
          ePICBodyCategory=DEFAULT
             ePICBodyType=DEFAULT
```

The object class for TraceServer is ePICApplication.

The DN is ePICAppId=TraceServer,o=ePICApplications,o=epic

The following elements and attributes are defined for this object:

ePICAppId
The name of the application.For example, *TraceServer*

ePICExceptionErrorLog
False

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICHostname
The name of the server that hosts this application.For example, *solmgr*

ePICLogging
False

Defines whether audit logging is performed.

ePICPropertyFile
A file that contains configuration parameters for this application.For example, *TraceServer.properties*

ePICTrace
False

Defines whether tracing is used.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

TraceServer ePICAppExtensions

The object class is ePICAppExtensions.

The DN is ePICAppExtensions,ePICAppId=TraceServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAppExtensions

ePICTraceHandler

Assigns the trace handler class that handles the trace.

For example, *com.ibm.logging.MultFileHandler*

ePICTraceMessageFile

Specifies the message catalog to use for traces. The *TraceClient* configuration uses the default catalog *com.ibm.epic.trace.client.TraceMessage*.

For example, *com.ibm.epic.trace.server.TraceServerMessage*

ePICTraceSyncOperation
False

Defines the message buffering technique to be used for trace messages (Synchronous writes or Asynchronous writes).False = Asynchronous.

TraceServer ePICAdapterRouting

The object class is ePICAdapterRouting.

The DN is cn=ePICAdapterRouting,ePICAppId=TraceServer,o=ePICApplications,o=epic

The following elements and attributes are defined for this object:

Common name
ePICAdapterRouting

Provides information about message types and the routing of messages.

ePICMQPPQueueMgr
Default

Defines the name of the queue manager when MQSeries is used as the communication transport mechanism.

ePICBodyCategory=DEFAULT

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,,ePICAppId=TraceServer, o=ePICApplications,o=epic

ePICBodyCategory
DEFAULT

Defines the body category of messages being sent..

TraceServer ePICBodyType=DEFAULT

The object class is ePICBodyType.

The DN is:

ePICBodyType=DEFAULT,ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting, ePICAppid=TraceServer,o=ePICApplications,o=epic

The following attributes are defined for this element:

ePICBodyType
DEFAULT

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
TraceServerAIQ

This is the name of the receive queue for this application.

Business Flow Manager

Business Flow Manager has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPId=BFM
      cn=ePICAppExtensions
```

The object class for BFM is ePICApplication.

The DN is ePICAppId=BFM,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId
BFM

The name of the application.

ePICExceptionErrorLog
False

Defines whether exception logging is performed.

ePICExceptionInfoLog

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICHostname

The name of the server that hosts this application.For example, *servlab04*

ePICLogging
False

Defines whether audit logging is performed.

ePICLogonInfoClassName
com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault

Defines the name of the logon class that is used to connect to the application when using an EAB adapter.

ePICPropertyFile

A file that contains configuration parameters for this application.For example, *BFM_ResourceBundle*

ePICTrace
False

Defines whether tracing is used.

ePICTraceClientID
TraceClient

Defines the name of the trace client configuration that is used. The default is *TraceClient*.

ePICTraceLevel
–1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level *–1* means all possible messages should be captured.

BFM ePICAppExtensions

The extension object class is ePICBFMExtensions.

The DN is ePICAppExtensions,ePICAppId=BFM,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAppExtensions

ePICAMSProviderURL
iiop://<ams_host_name>:900

ePICMQWFAdminPassword
password

ePICMQWFAdminUserId
ADMIN

ePICMQWFGSOTargetResource
MQWorkflow

ePICWASWorkflowProviderURL
iiop://<waswf_host_name>:900

ePICWWFEvents
False

ePICWWFProfile
False

LogErrorQ

LogErrorQ has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPId=ePICLogErrorQ
      cn=ePICAdapterRouting
          ePICBodyCategory=DEFAULt
             ePICBodyType=DEFAULT
```

The object class for ePICLogErrorQ is ePICApplication.

The DN is ePICAppId=ePICLogErrorQ,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId
ePICLogErrorQ

The name of the application.

ePICExceptionErrorLog
True

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICLogging
True

Defines whether audit logging is performed.

ePICTrace
True

Defines whether tracing is used.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

LogErrorQ ePICAdapterRouting

The object class is ePICAdapterRouting.

The DN is:
cn=ePICAdapterRouting,,ePICAppId=ePICLogErrorQ,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAdapterRouting

Provides information about message types and the routing of messages.

ePICMQPPQueueMgr
Default

Defines the name of the queue manager when MQSeries is used as the communication transport mechanism.

LogErrorQ ePICBodyCategory=DEFAULT

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,ePICAppId=LogErrorQ, o=ePICApplications,o=epic

ePICBodyCategory
DEFAULT

Defines the body category of messages being sent.

LogErrorQ ePICBodyType=DEFAULT

The object class is ePICBodyType.

The DN is:

ePICBodyType=DEFAULT,ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting, ePICAppId=LogErrorQ,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
DEFAULT

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
TraceServerAIQ

This is the name of the receive queue for this application.

EventService

Event service has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=EventService
      cn=ePICAppExtensions
          ePICEventServer=DEFAULT
```

The object class for the event service is ePICApplication.

The DN is ePICAppid=EventService,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId
EventService

The name of the application.

ePICExceptionErrorLog
True

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICLogging
False

Defines whether audit logging is performed.

ePICTrace
False

Defines whether tracing is used.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

EventService ePICAppExtensions

The extension object class is ePICAppExtensions.

The DN is
ePICAppExtensions,ePICAppId=EventService,o=ePICApplications,o=epic

ePICEventServiceExtensions Attributes: The following attributes are defined for this object:

Common name
ePICAppExtensions

ePICConstraintsURL
http://servlab02.bocaraton.ibm.com/epic/xml/consdef.xml

ePICEventsDTDURL
http://servlab02.bocaraton.ibm.com/epic/xml/events.dtd

ePICEventsURL
http://servlab02.bocaraton.ibm.com/epic/xml/evdef.xml

ePICSubscriptionURL
http://servlab02.bocaraton.ibm.com/epic/xml/subsciption.xml

EventService ePICEventServer=Default: The object class is ePICEventServer.

The DN is:

ePICEventServer=DEFAULT,cn=cPICAppExtensions,ePICAppId=EventService,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICEventServer
DEFAULT

ePICMQChannel
java.channel

ePICMQPPQueueMgr
QM_servlab02

Defines the name of the queue manager when MQSeries is used as the communication transport mechanism.

ePICPortNumber
1515

host *servlab02.bocaraton.ibm.com*

Trust and Access Manager

The Trust and Access Manager has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=AMS
      cn=ePICAppExtensions
      cn=ePICAdapterRouting
          ePICBodyCategory=DEFAULT
             ePICBodyType=DEFAULT
      cn=ePICBodyCategory=ePICMessage
             ePICBodyType=ePICExceptionLog
```

The object class for AMS is ePICApplication.

The DN is ePICAppId=AMS,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId
AMS

The name of the application.

ePICExceptionErrorLog
True

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICLogging
False

Defines whether audit logging is performed.

ePICTrace
False

Defines whether tracing is used.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

AMS ePICAppExtensions

The extension object class is ePICAppExtensions.

The DN is ePICAppExtensions,ePICAppId=AMS,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAppExtensions

ePICAMSPDUsername
ams_user

Defines the name of the AMS user that is using Policy Director.

ePICAMSUserRegistry
PD

Defines the user registry to be used by AMS. Allowable values are **PD** or **LDAP**.

ePICLDAPBindPassword
password

Defines the GSO user password.

ePICLDAPBindUserDN
cn=root

Defines the GSO user distinguished name. The default is *AMS_GSO_User*

ePICLDAPServer
Blank

Defines the LDAP server that is used for global sign on.

ePICPersonNamingAttribute
uid

Defines the attribute that is used to access a user entry in LDAP. The default is *uid*

ePICUserRegistryDN
o=ePICUsers,o=ePIC

Defines the distinguished name where the User Registry starts in the LDAP directory.

userPassword *sec_master*

Defines the password of the Policy Director user.

AMS ePICAdapterRouting

The object class is ePICAdapterRouting.

The DN is:

cn=ePICAdapterRouting,ePICAppExtensions,ePICAppId=AMS,
o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
: *ePICAdapterRouting*

Provides information about message types and the routing of messages.

ePICMQPPQueueMgrPortNumber

Specifies the port number of the server process of the queue manager. The default is 1414.

AMS ePICBodyCategory=DEFAULT

The object class is ePICBodyCategory.

The DN is

ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,ePICAppExtensions,
ePICAppId=AMS,o=ePICApplications,o=epic

ePICBodyCategory
DEFAULT

Defines the body category of messages being sent.

AMS ePICBodyType=DEFAULT: The object class is ePICBodyType. The DN is

ePICBodyType=DEFAULT,ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,
ePICAppExtensions,ePICAppId=AMS,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
DEFAULT

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
ePICExceptionServer

This is the name of the receive queue for this application.

ePICReceiveTimeout
30000

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of –1 specifies the receiver waits indefinitely.

AMS ePICBodyCategory=ePICMessage

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=ePICMessage,cn=ePICAdapterRouting,ePICAppExtensions, ePICAppId=AMS,o=ePICApplications,o=epic

ePICBodyCategory
ePICMessage

Defines the body category of messages being sent.

AMS ePICBodyType=ePICExceptionLog

The object class is ePICBodyType.

The DN is:

ePICBodyType=ePICExceptionLog,ePICBodyCategory=ePICMessage, cn=ePICAdapterRouting,ePICAppExtensions,ePICAppId=AMS, o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
ePICExceptionLog

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
ePICExceptionServer

This is the name of the receive queue for this application.

ePICReceiveTimeout

30000

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of −1 specifies the receiver waits indefinitely.

serverEventInQ

serverEventInQ has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=serverEventInQ
      cn=ePICAdapterRouting
          ePICBodyCat
             ePICBodyType=DEFAULT
```

The object class for serverEventInQ is ePICApplication.

The DN is ePICAppid=serverEventInQ,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppid
serverEventInQ

The name of the application.

ePICExceptionErrorLog
True

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICLogging
True

Defines whether audit logging is performed.

ePICTrace
True

Defines whether tracing is used.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

serverEventInQ ePICAdapterRouting

The object class is ePICAdapterRouting.

The DN is:

cn=ePICAdapterRouting,ePICAppId=serverEventInQ,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAdapterRouting

Provides information about message types and the routing of messages.

ePICMQPPQueueMgr
DEFAULT

Defines the name of the queue manager when MQSeries is used as the communication transport mechanism.

serverEventInQ ePICBodyCategory=DEFAULT

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,ePICAppId=serverEventInQ,o=ePICApplications,o=epic

ePICBodyCategory
DEFAULT

Defines the body category of messages being sent.

serverEventInQ ePICBodyType=DEFAULT

The object class is ePICBodyType.

The DN is:

ePICBodyType=DEFAULT,ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,ePICAppId=serverEventInQ,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
DEFAULT

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
serverEventInQ

This is the name of the receive queue for this application.

logServerControl

logServerControl has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=logServerControl
      cn=ePICAdapterRouting
          ePICBodyCategory=DEFAULT
             ePICBodyType=DEFAULT
```

The object class for logServerControl is ePICApplication.

The DN is: ePICAppid=logServerControl,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId *logServerControl*

ePICExceptionErrorLog
True

Defines whether exception logging is performed.

ePICExceptionInfoLog
True

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
True

Defines whether warning errors are logged.

ePICLogging True

Defines whether audit logging is performed.

ePICTrace True

Defines whether tracing is used.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

logServerControl ePICAdapterRouting

The object class is ePICAdapterRouting.

The DN is:
cn=ePICAdapterRouting,ePICAppId=logServerControl,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAdapterRouting

Provides information about message types and the routing of messages.

ePICMQPPQueueMgr
DEFAULT

Defines the name of the queue manager when MQSeries is used as the communication transport mechanism.

logServerControl ePICBodyCategory=DEFAULT

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,ePICAppId=logServerControl,
o=ePICApplications,o=epic

ePICBodyCategory
DEFAULT

Defines the body category of messages being sent.

logServerControl ePICBodyType=DEFAULT

The object class is ePICBodyType.

The DN is:

ePICBodyType=DEFAULT,ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,
ePICAppId=logServerControl,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
DEFAULT

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
ePICLogServerEvent

This is the name of the receive queue for this application.

ePICReceiveTimeout
30000

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of –1 specifies the receiver waits indefinitely.

Interaction Manager

Interaction Manager has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=IM
      cn=ePICAppExtensions
```

The object class for IM is ePICApplication.

The DN is ePICAppId=IM,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId
IM

The name of the application.

ePICExceptionErrorLog
True

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
Not specified

Defines whether warning errors are logged.

ePICHostname

The name of the server that hosts this application.For example, *servlab08*

ePICLogging
False

Defines whether audit logging is performed.

ePICTrace
False

Defines whether tracing is used.

ePICTraceClientID
TraceClient

Defines the name of the trace client configuration that is used. The default is *TraceClient*.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

IM ePICAppExtensions

The extension object class is ePICIMExtensions.

The DN is ePICAppExtensions,ePICAppId=IM,o=ePICApplications,o=epic

ePICIMExtensions Attributes: The following attributes are defined for this object:

Common name
ePICAppExtensions

ePICAMSProviderURL
iiop://servlab03:900

ePICBusinessServicesProviderURL
iiop://<bizsvc_host_name>:900

ePICWorkflowServicesProviderURL
iiop://servlab04:900

TraceClient

TraceClient has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=TraceClient
      cn=ePICAppExtensions
          ePICTraceHandler=com.ibm.logging.ConsoleHandler
          ePICTraceHandler=com.ibm.logging.FileHandler
          ePICTraceHandler=com.ibm.logging.ENAHandler
          ePICTraceHandler=com.ibm.logging.SocketHandler
```

The object class for TraceClient is ePICApplication.

The DN is ePICAppId=TraceClient,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId
TraceClient

The name of the application.

ePICExceptionErrorLog
False

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICTrace
False

Defines whether tracing is used.

ePICTraceLevel
1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level 1 means Information Only messages are captured.

TraceClient ePICAppExtensions

The object class is ePICTraceExtensions.

The DN is
ePICAppExtensions,ePICAppId=TraceClient,o=ePICApplications,o=epic

ePICTraceExtensions Attributes: The following attributes are defined for this object:

Common name
: *ePICAppExtensions*

ePICDepAppID
: *TraceServer*

ePICTraceHandler
: *com.ibm.logging.FileHandler*

  Assigns the trace handler class that handles the trace.

ePICTraceMessageFile
: *com.ibm.epic.trace.client.TraceMessage*

  Specifies the message catalog to use for traces. The *TraceClient* configuration uses the default catalog *com.ibm.epic.trace.client.TraceMessage*.

ePICTraceSyncOperation
: True

  Defines the message buffering technique to be used (Synchronous writes or Asynchronous writes). True = Synchronous

Trace Client Console ePICTraceHandler

The object class is ePICTraceHandler.

The DN is:

ePICTraceHandler=com.ibm.logging.ConsoleHandler,cn=ePICAppExtensions,
ePICAppId=TraceClient,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICTraceHandler
: *com.ibm.logging.ConsoleHandler*

  Assigns the trace handler class that handles the trace.

ePICTraceFormatter
: *com.ibm.epic.trace.client.EpicTraceFormatter*

  Specifies the formatter that is used for this trace. The default trace file is trc.log and has no limit on the size of the file.

Trace Client File ePICTraceHandler

The object class is ePICTraceHandler.

The DN is:

ePICTraceHandler=com.ibm.logging.FileHandler,cn=ePICAppExtensions,
ePICAppId=TraceClient,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICTraceHandler
: *com.ibm.logging.FileHandler*

Assigns the trace handler class that handles the trace.

ePICTraceFileName
: *c:\epic\logs\ePIC.trc*

ePICTraceFormatter
: *com.ibm.epic.trace.client.EpicTraceFormatter*

Specifies the formatter that is used for this trace. The default trace file is trc.log and has no limit on the size of the file.

Trace Client ENA ePICTraceHandler

The object class is ePICTraceHandler.

The DN is:

ePICTraceHandler=com.ibm.logging.ENAHandler,cn=ePICAppExtensions,
ePICAppId=TraceClient,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICTraceHandler
: *com.ibm.logging.ENAHandler*

Assigns the trace handler class that handles the trace.

ePICTraceFormatter
: *com.ibm.epic.trace.client.EpicXMLFormatter*

Assigns the trace handler class that handles the trace. ENA handlers forward trace messages to a trace server through a socket connection or a native adapter. For additional information about trace handlers refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897.

Trace Client Socket ePICTraceHandler

The object class is ePICTraceHandler.

The DN is:

ePICTraceHandler=com.ibm.logging.SocketHandler,cn=ePICAppExtensions, ePICAppId=TraceClient,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICTraceHandler
: *com.ibm.logging.SocketHandler*

  Assigns the trace handler class that handles the trace.

ePICTraceFormatter
: *com.ibm.epic.trace.client.EpicXMLFormatter*

  Specifies the formatter that is used for this trace. The default trace file is trc.log and has no limit on the size of the file.

ePICLogServer

LogServer has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=ePICLogServer
      cn=ePICAdapterRouting
          ePICBodyCategory=DEFAULT
             ePICBodyType=DEFAULT
          ePICBodyCategory=ePICMESSAGE
             ePICBodyType=ReadRequest
               cn=AuditLogRead
```

The object class for ePICLogServer is ePICApplication.

The DN is ePICAppId=ePICLogServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId
ePICLogServer

The name of the application.

ePICExceptionErrorLog
True

Defines whether exception logging is performed.

ePICExceptionInfoLog
True

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
True

Defines whether warning errors are logged.

ePICLogging
False

Defines whether audit logging is performed.

ePICTrace
False

Defines whether tracing is used.

ePICTraceClientID
TraceClient

Defines the name of the trace client configuration that is used. The default is *TraceClient*.

ePICTraceLevel
1835008

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897.

LogServer ePICAdapterRouting
The object class is ePICAdapterRouting.

The DN is
cn=ePICAdapterRouting,,ePICAppId=ePICLogServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAdapterRouting

Provides information about message types and the routing of messages.

ePICMQPPQueueMgr
DEFAULT

Defines the name of the queue manager when MQSeries is used as the communication transport mechanism.

LogServer ePICBodyCategory=DEFAULT
The object class is ePICBodyCategory.

The DN is

ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,ePICAppId=LogServer,
o=ePICApplications,o=epic

ePICBodyCategory
DEFAULT

Defines the body category of messages being sent.

LogServer ePICBodyType=DEFAULT
The object class is ePICBodyType.

The DN is:

ePICBodyType=DEFAULT,ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,
ePICAppId=LogServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
DEFAULT

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
epic_Log_Server

This is the name of the receive queue for this application.

ePICReceiveTimeout
900000

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of –1 specifies the receiver waits indefinitely.

LogServer ePICBodyCategory=ePICMessage

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=ePICMessage,cn=ePICAdapterRouting,ePICAppId=LogServer, o=ePICApplications,o=epic

ePICBodyCategory
ePICMessage

Defines the body category of messages being sent.

LogServer ePICBodyType=ReadRequest

The object class is ePICBodyType.

The DN is:

ePICBodyType=ReadRequest,ePICBodyCategory=ePICMessage, cn=ePICAdapterRouting,ePICAppId=LogServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
ReadRequest

Defines the body type of messages being sent.

ePICCommandClassName
com.ibm.epic.log.server.AuditWorker

Specifies the name of an EAB target adapter or EJB command that is invoked to process messages.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
AuditLogRead

This is the name of the receive queue for this application.

ePICReceiveTimeout
999999999

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of −1 specifies the receiver waits indefinitely.

AuditLogRead: The object class is javaContainer.

The DN is:

cn=AuditLogRead,ePICBodyType=ReadRequest,ePICBodyCategory=ePICMessage, cn=ePICAdapterRouting,ePICAppId=ePICLogServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
AuditLogRead

javaClassName
com.ibm.mq.jms.MQQueue

This is the class name of the Java Message Service queue manager that receives and sends messages.

javaFactory *com.ibm.mq.jms.MQQueueFactory*

This is the Java Message Service queue connection factory that receives and sends messages.

javaReferenceAddress
#0#VER#1

Exception Server

ExceptionServer has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=ePICExceptionServer
      cn=ePICAdapterRouting
          ePICBodyCategory=DEFAULT
             ePICBodyType=DEFAULT
          ePICBodyCategory=ePICMessage
             ePICBodyType=ePICXmlExceptionLog
             ePICBodyType=ePICExceptionLog
      cn=ePICAppExtensions
          ePICSolMgrGroup=Configuration
             ePICSolMgrGroupMember=URL
             ePICSolMgrGroupMember=KEY
             ePICSolMgrGroupMember=BUFFER_SIZE
             ePICSolMgrGroupMember=USER
             ePICSolMgrGroupMember=UPGRADEPOLICY
             ePICSolMgrGroupMember=ConsoleTimeOutmnts
             ePICSolMgrGroupMember=SELECT
             ePICSolMgrGroupMember=UPDATE
          ePICSolMgrGroup=DisplayConsole
             ePICSolMgrGroupMember=Severity
             ePICSolMgrGroupMember=Time Stamp
          ePICSolMgrGroup=AppDomain
             ePICSolMgrGroupMember=ExceptionID
             ePICSolMgrGroupMember=ExceptionTypeID
             ePICSolMgrGroupMember=ExceptionSource
          ePICSolMgrGroup=SolMgrConsole
             ePICSolMgrGroupMember=ExceptionLog
             ePICSolMgrGroupMember=AuditLog
             ePICSolMgrGroupMember=AuditLogCannedSearchServlet
                ePICSolMgrGroupSubMember=PO
                ePICSolMgrGroupSubMember=USER
             ePICSolMgrGroupMember=LdapConfigServlet
             ePICSolMgrGroupMember=TraceDisplayServlet
          ePICSolMgrGroup=ExceptionTYpe
```

The object class for ExceptionServer is ePICApplication.

The DN is ePICAppId=ExceptionServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId *Exception Server*

The name of the application.

ePICExceptionErrorLog
False

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICLogging
False

Defines whether audit logging is performed.

ePICPropertyFile

A file that contains configuration parameters for this application.For example, *EM.conf*

ePICTrace
False

Defines whether tracing is used.

ePICTraceClientID
TraceClient

Defines the name of the trace client configuration that is used. The default is *TraceClient*.

ePICTraceLevel
–1

Defines the tracing level. For additional information about trace levels refer to the *IBM MQSeries® Adapter Kernel For Multiplatforms Problem Determination Guide*, GC34-5897. Trace Level *–1* indicates all possible messages should be captured.

ExceptionServer ePICAdapterRouting

The object class is ePICAdapterRouting.

The DN is:

cn=ePICAdapterRouting,ePICAppExtensions,ePICAppId=ExceptionServer, o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAdapterRouting

Provides information about message types and the routing of messages.

ePICMQPPQueueMgr
DEFAULT

Defines the name of the queue manager when MQSeries is used as the communication transport mechanism.

ExceptionServer ePICBodyCategory=DEFAULT

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,
ePICAppId=ExceptionServer,o=ePICApplications,o=epic

ePICBodyCategory
DEFAULT

Defines the body category of messages being sent.

ExceptionServer ePICBodyType=DEFAULT

The object class is ePICBodyType.

The DN is:

ePICBodyType=DEFAULT,ePICBodyCategory=DEFAULT,cn=ePICAdapterRouting,
AppId=ExceptionServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
DEFAULT

Defines the body type of messages being sent.

ePICReceiveTimeOut
30000

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of −1 specifies the receiver waits indefinitely.

ExceptionServer ePICBodyCategory=ePICMessage

The object class is ePICBodyCategory.

The DN is:

ePICBodyCategory=ePICMessage,cn=ePICAdapterRouting,
ePICAppId=ePICExceptionServer,o=ePICApplications,o=epic

ePICBodyCategory
ePICMessage

Defines the body category of messages being sent.

ExceptionServer ePICBodyType=ePICXMLExceptionLog

The object class is ePICBodyType.

The DN is:

ePICBodyType=ePICXMLExceptionLog,ePICBodyCategory=ePICMessage,
cn=ePICAdapterRouting,ePICAppId=ExceptionServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
ePICXmlExceptionLog

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
ePICXmlExceptionServer

This is the name of the receive queue for this application.

ePICReceiveTimeOut
30000

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of −1 specifies the receiver waits indefinitely.

ExceptionServer ePICBodyType=ePICExceptionLog

The object class is ePICBodyType.

The DN is:

ePICBodyType=ePICExceptionLog,ePICBodyCategory=ePICMessage,
cn=ePICAdapterRouting,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
ePICExceptionLog

Defines the body type of messages being sent.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
ePICExceptionServer

This is the name of the receive queue for this application.

ePICReceiveTimeOut
30000

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of −1 specifies the receiver waits indefinitely.

ExceptionServer ePICBodyType=ReadRequest

The object class is ePICBodyType.

The DN is:

ePICBodyType=ReadRequest,ePICBodyCategory=ePICMessage,
cn=ePICAdapterRouting,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICBodyType
ReadRequest

Defines the body type of messages being sent.

ePICCommandClassName
com.ibm.epic.log.server.ExceptionWorker

Specifies the name of an EAB target adapter or EJB command that is invoked to process messages.

ePICReceiveMode
MQPP

Defines the transport mechanism as MQSeries base services.

ePICReceiveMQPPQueue
ExceptionLogRead

This is the name of the receive queue for this application.

ePICReceiveTimeOut
999999

Specifies, in milliseconds, the length of time the receiver waits for messages before it times out. A value of –1 specifies the receiver waits indefinitely.

ExceptionServer ePICAppExtensions

The object class is ePICAdapterDaemonExtensions.

The DN is
ePICAppExtensions,ePICAppId=ExceptionServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

Common name
ePICAppExtensions

ePICMaxWorkers
5

ePICMinWorkers
1

ExceptionServer SolMgrGroup=Configuration

The object class is ePICSolMgrGroup.

The DN is:

ePICSolMgrGroup,cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

This object class has one defined attribute: **ePICSolMgrGroup** = Configuration.

ExceptionServer ePICSolMgrGroupMember=URL

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=URL,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
URL

validValues *jdbc:db2:EM*

ExceptionServer ePICSolMgrGroupMember=KEY

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=KEY,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
KEY

validValues *db2admin*

ExceptionServer ePICSolMgrGroupMember=BUFFER_SIZE

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=BUFFER_SIZE,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,o=ePICApplications,
o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
BUFFER_SIZE

validValues *3*

ExceptionServer ePICSolMgrGroupMember=USER

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=USER,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
USER

validValues *db2admin*

ExceptionServer ePICSolMgrGroupMember=UPGRADEPOLICY

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=UPGRADEPOLICY,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
UPGRADEPOLICY

validValues *Update*

ExceptionServer ePICSolMgrGroupMember=ConsoleTimeOutmnts

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=ConsoleTimeOutmnts,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,o=ePICApplications,
o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
ConsoleTimeOutmnts

description Solution Console time out ***IN minutes***

validValues *1*

ExceptionServer ePICSolMgrGroupMember=SELECT

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=SELECT,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
SELECT

validValues *c:\\epic\\xml\\exceptiondataretrieve_new.xml*

ExceptionServer ePICSolMgrGroupMember=UPDATE

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=UPDATE,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
UPDATE

validValues *c:\\epic\\xml\\daoinsertqueries.xml*

Exception Server ePICSolMgrGroup=Display Console

The object class is ePICSolMgrGroup.

The DN is:

ePICSolMgrGroup=DisplayConsole,cn=ePICAppExtensions,
ePICAppId=ePICExceptionServer,o=ePICApplications,o=epic

This following attributes are defined for this object:

ePICSolMgrGroup
DisplayConsole

description Exception Manager console configuration

ExceptionServer ePICSolMgrGroupMember=Severity

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=Severity,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
Severity

validValues *PRIORITY*

ExceptionServer ePICSolMgrGroupMember=Time Stamp

The object class is ePICSolMgrGroupMember.

The DN is

ePICSolMgrGroupMember=Time Stamp,ePICSolMgrGroup=Configuration,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
Time Stamp

validValues *EXCEPTIONTIME*

This object provides the date and time of an exception in the format: MM:DD:YYYY:HH:MM:SS where:

MM = 01 through 12
DD = 01 through 31
YYYY = 0001 through 9999
HH = 00 through 24
MM = 00 through 59
SS = 00 through 59

Exception Server ePICSolMgrGroup=AppDomain

The object class is ePICSolMgrGroup.

The DN is:

ePICSolMgrGroup=AppDomain,cn=ePICAppExtensions,
ePICAppId=ePICExceptionServer,o=ePICApplications,
o=epic

This following attributes are defined for this object:

ePICSolMgrGroup
AppDomain

description AppDomain

ExceptionServer ePICSolMgrGroupMember=ExceptionID

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=ExceptionID,ePICSolMgrGroup=AppDomain,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
EXCEPTIONID

validValues *EXCEPTIONID*

ExceptionServer ePICSolMgrGroupMember=ExceptionTypeID

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=ExceptionTypeID,ePICSolMgrGroup=AppDomain,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
ExceptionTypeID

validValues *EXCEPTIONTYPEID*

ExceptionServer ePICSolMgrGroupMember=ExceptionSource

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=ExceptionSource,ePICSolMgrGroup=AppDomain,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
ExceptionSource

validValues *EXCEPTIONSOURCE*

Exception Server ePICSolMgrGroup=SolMgrConsole

The object class is ePICSolMgrGroup.

The DN is:

ePICSolMgrGroup=SolMgrConsole,cn=ePICAppExtensions,
ePICAppId=ePICExceptionServer,o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroup
SolMgrConsole

ExceptionServer ePICSolMgrGroupMember=ExceptionLog

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=ExceptionLog,ePICSolMgrGroup=SolMgrConsole,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
ExceptionLog

validValues *DisplayExceptionLogServlet*

ExceptionServer ePICSolMgrGroupMember=AuditLog

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=AuditLog,ePICSolMgrGroup=SolMgrConsole,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
AuditLog

validValues *DisplayAuditLogServlet*

ExceptionServer ePICSolMgrGroupMember=AuditLogCannedSearchServlet

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=AuditLogCannedSearchServlet,
ePICSolMgrGroup=SolMgrConsole,cn=ePICAppExtensions,
ePICAppId=ePICExceptionServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
AuditLogCannedSearchServlet

validValues *DisplayAuditLogCannedSearchServlet*

ExceptionServer ePICSolMgrGroupSubMember=PO: The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupSubMember=PO,ePICSolMgrGroupMember=
AuditLogCannedSearchServlet,ePICSolMgrGroup=SolMgrConsole,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupSubMember
PO

validValues *DisplayAuditLogCannedSearchServlet?group=PO*

ExceptionServer ePICSolMgrGroupSubMember=USER: The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupSubMember=USER,ePICSolMgrGroupMember=
AuditLogCannedSearchServlet,ePICSolMgrGroup=SolMgrConsole,
cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupSubMember
USER

validValues *DisplayAuditLogCannedSearchServlet?group=USER*

ExceptionServer ePICSolMgrGroupMember=LdapConfigServlet

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=LdapConfigServlet,ePICSolMgrGroup=
SolMgrConsole,cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
LdapConfigServlet

validValues *DisplayLDAPConfigServlet*

ExceptionServer ePICSolMgrGroupMember=TraceDisplayServlet

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=TraceDisplayServlet,ePICSolMgrGroup=SolMgrConsole,cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
TraceDisplayServlet

validValues *TraceDispPickServlet*

Exception Server ePICSolMgrGroup=ExceptionType

The object class is ePICSolMgrGroup.

The DN is:

ePICSolMgrGroup=ExceptionType,cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,o=ePICApplications,o=epic

This following attributes are defined for this object:

ePICSolMgrGroup
ExceptionType

description Type of exceptions registered in the Exception Manager

ExceptionServer ePICSolMgrGroupMember=ExceptionLog

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=ExceptionLog,ePICSolMgrGroup=SolMgrConsole,cn=ePICAppExtensions,ePICAppId=ePICExceptionServer,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
ExceptionLog

validValues *DisplayExceptionLogServlet*

ePICAuditCannedSearch

AuditCannedSearch has the following directory structure:

```
o=epic
  o=ePICApplications
    ePICAPPID=ePICAuditCannedSearch
      cn=ePICAppExtensions
          ePICSolMgrGroup=PO
             ePICSolMgrGroupMember=Query
             ePICSolMgrGroupMember=Columns
                ePICSolMgrGroupSubMember=State
                ePICSolMgrGroupSubMember=CustPoId
                ePICSolMgrGroupSubMember=Timestamp
          ePICSolMgrGroup=USER
             ePICSolMgrGroupMember=QUERY
             ePICSolMgrGroupMember=Columns
                ePICSolMgrGroupSubMember=UserId
          ePICSolMgrGroup=System
             ePICSolMgrGroupMember=QUERY
             ePICSolMgrGroupMember=Columns
                ePICSolMgrGroupSubMember=TransactionID
                ePICSolMgrGroupSubMember=SourceMsg
                ePICSolMgrGroupSubMember=UniqueId
```

The object class for AuditCannedSearch is ePICApplication.

The DN is ePICAppId=ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICAppId *ePICAuditCannedSearch*

The name of the application.

ePICExceptionErrorLog
False

Defines whether exception logging is performed.

ePICExceptionInfoLog
False

Defines whether exception logging for informational messages is performed.

ePICExceptionWarningLog
False

Defines whether warning errors are logged.

ePICLogging False

Defines whether audit logging is performed.

ePICTrace False

Defines whether tracing is used.

AuditCannedSearch ePICAppExtensions

The object class is ePICAgentExtensions.

The DN is
ePICAppExtensions,ePICAppId=ePICAuditCannedSearch,o=ePICApplications,o=epic

This object has one defined attribute: **Common name** = *ePICAppextension*

SolMgrGroup=PO

The object class is ePICSolMgrGroup.

The DN is:

ePICSolMgrGroup=PO,cn=ePICAppExtensions,ePICAppId=ePICAuditCannedSearch,
o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroup
PO

description
select msg_xaction_id,log_time_stamp from epic2

ePICSolMgrGroupMember=Query

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=Query,ePICSolMgrGroup=PO,cn=ePICAppExtensions,
ePICAppId=ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
Query

validValues
Select state,msg_time_stamp,custpoid from epic_xml_view where msg_xaction_id=

ePICSolMgrGroupMember=Columns

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=Columns,ePICSolMgrGroup=PO,cn=ePICAppExtensions,
ePICAppId=ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroupMember
Columns

SolMgrGroup=USER

The object class is ePICSolMgrGroup.

The DN is:

ePICSolMgrGroup=USER,cn=ePICAppExtensions,ePICAppId=ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroup
USER

description select msg_uniqueid from epic2

ePICSolMgrGroupMember=Query

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=Query,ePICSolMgrGroup=USER,cn=ePICAppExtensions,ePICAppId=ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
Query

validValues *Select src_msg from EPIC 2 where msg_unique_id=*

ePICSolMgrGroupMember=Columns

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=Columns,ePICSolMgrGroup=USER,cn=ePICAppExtensions,ePICAppId=ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroupMember
Columns

SolMgrGroupSubmember=UserId

The object class is ePICSolMgrGroupSubMember.

The DN is:

ePICSolMgrGroupSubMember=UserId,ePICSolMgrGroupMember=Columns, ePICSolMgrGroup=USER,cn=ePICAppExtensions,ePICAppId= ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroupSubMember
: *UserId*

SolMgrGroup=System

The object class is ePICSolMgrGroup.

The DN is

ePICSolMgrGroup=System,cn=ePICAppExtensions,ePICAppId= ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroup
: *System*

description
: select msg_xaction_id,msg_time_stamp from epic2

ePICSolMgrGroupMember=Query

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=Query,ePICSolMgrGroup=System, cn=ePICAppExtensions,ePICAppId=ePICAuditCannedSearch, o=ePICApplications,o=epic

The following attributes are defined for this object:

ePICSolMgrGroupMember
: *Query*

validValues
: *Select msg_xaction_id.src_msg,msg_unique_id from EPIC 2 where msg_xaction_id=*

ePICSolMgrGroupMember=Columns

The object class is ePICSolMgrGroupMember.

The DN is:

ePICSolMgrGroupMember=Columns,ePICSolMgrGroup=System,
cn=ePICAppExtensions,ePICAppId=ePICAuditCannedSearch,
o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroupMember
Columns

SolMgrGroupSubmember=TransactionId

The object class is ePICSolMgrGroupSubMember.

The DN is:

ePICSolMgrGroupSubMember=TransactionId,ePICSolMgrGroupMember=Columns,
ePICSolMgrGroup=System,cn=ePICAppExtensions,ePICAppId=
ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroupSubMember
TransactionID

SolMgrGroupSubmember=SourceMsg

The object class is ePICSolMgrGroupSubMember.

The DN is:

ePICSolMgrGroupSubMember=SourceMsg,ePICSolMgrGroupMember=Columns,
ePICSolMgrGroup=System,cn=ePICAppExtensions,ePICAppId=
ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroupSubMember
SourceMsg

SolMgrGroupSubmember=UniqueId

The object class is ePICSolMgrGroupSubMember.

The DN is:

ePICSolMgrGroupSubMember=UniqueId,ePICSolMgrGroupMember=Columns,
ePICSolMgrGroup=System,cn=ePICAppExtensions,ePICAppId=
ePICAuditCannedSearch,o=ePICApplications,o=epic

The following attribute is defined for this object:

ePICSolMgrGroupSubMember
UniqueId

Solutions Tree (ePICSolutions)

The solutions and awarelets tree contains directory entries for Business Integrator solutions and awarelets. This section provides representative LDAP entries for Solutions that could be deployed in Business Integrator runtime environment. The LDAP for an actual Solution could contain similar entries.

ePICSolutions Sample

The object class for a solutions is organization.

The DN is: o=ePICSolutions,o=epic

The following attributes can be defined for this object:

0 *ePICSolutions*

businessCategory

description This attribute can be used to provide a description of what services this Solution provides or describe its business purpose.

destinationIndicator

fascimileTelephoneNumber

internationalISDNNumber

I

Office phone:

PhysicalDeliveryOfficeName

postalAddress

postalCode

registeredAddress

searchGuide

seeAlso

st

street

teletexTerminalIdentifier

telexNumber

userPassword

x121Address

cn=Solution1

The object class for Solutions1 is ePICSolution.

The DN is: cn=Solution1,o=ePICSolutions,o=epic

The following attributes are defined for this object:

Common name:
Solution1

ePICSolutionEntry
This is name and location of the servlet associated with this solution. For example, *servlet**com.ibm.epic.solution.solution1Solution1Servlet*

description
This attribute is provided for descriptive purposes, such as identifying the name and purpose of the solution.

ePICListofRoles
This attribute identifies the User Roles that can access this solution. For example *Buyer, Vendor*.

ePICPDObjectSpace
The prefix used to build a Policy Director object, for example, */BiTemplates/BtoBi*. This name is used by AMS during initialization.

ePICProcessDefinitions

ePICTargetFrame
Fame2

ePICWorkFlowServlet
This is the name and location of the WorkFlow servlet associated with this solution.

Awarelets

The object class for Awarelets is organization

The DN is: o=Awarelets,cn=Solution1,o=ePICSolutions,o=epic

Awarelets use the same set of attributes as ePICSolutions objects. See “ePICSolutions Sample” on page 133 for a description of these attributes.

cn=WorkflowAwarelet

The object class is ePICAwarelet

The DN is:

cn=WorkflowAwarelets,o=Awarelets,cn=Solution1,o=ePICSolutions,
o=epic

The following attributes can be defined for this object:

Common name
WorkFlowAwarelet

ePICAssignments
EpicSystem

ePICAwareletMasks
MASK5

ePICClassName
com.ibm.epic.im.portal.awarelets.WorkflowAwarelet

cn=ePICSystem

The object class is ePICAssignment

The DN is:

cn=ePICSystem,cn=WorkflowAwarelets,o=Awarelets,cn=Solution1,
o=ePICSolutions,o=epic

The following attributes can be defined for this object:

Common name
EpicSystem

ePICAwareletMasks
MASK5

cn=BtoBitemplates

The following templates are provided under this solution object. In the sample templates provided are News Bulletin objects that can be targeted to specific areas of a business such as Administration and Accounting. This provides a mechanism to target only that news that has business value to specific users. This not only saves network time but serves to save employee time as they would not have to sort through News items that have no relationship to their work assignments.

The next sections show the objects that are used to generate a news bulletin to a supplier.

NewsBulletins

The object class is organization.

The DN is:

o=NewsBullentins,cn=BtoBiTemplates,o=ePICSolutions,o=epic

The only defined attribute is **o** = *NewsBulletins*.

The remaining attributes that can be defined are described in “ePICSolutions Sample” on page 133.

ePICNewsBullentinID=SupplierNews01152001: The object class is ePICNewsBulletin.

The DN is:

ePICNewsBulletinID=SupplierNews01152001_1,o=NewsBulletins,
n=BtoBiTemplates,o=ePICSolutions,o=epic

The following attributes are defined for this object:

ePICCategory
The Category assigned is *SupplierNews*

ePICDate
The date of the news item is *January 15, 2001*

ePICNewsBullentID
The Unique ID for this bulletin is *SupplierNews01152001_1*

ePICNewsTitle
The title label for this news item is *Volume discounts on seat covers expires March 15, 2001*

url
The universal resource locator including the news bulletin file name is */ePortal/html/SupplierNews01152001_1.html*.

In addition to the templates for news items, templates for menu items that are rendered on users desktops are provided. These items can be targeted to specific users and specific roles, for system management components. Essentially a reusable object can be defined that can be reused when needed.

The next sections show the objects that are used to provide the Exception Console gif menu item.

MenuItem

The object class is organization.

The DN is:

o=MenuItem,cn=BtoBiTemplates,o=ePICSolutions,o=epic

The only defined attribute is **o** = *MenuItem*.

The remaining attributes that can be defined are described in "ePICSolutions Sample" on page 133.

ePICMenuItem=ExceptionConsole: The object class is ePICMenuItem.

The DN is:

ePICMenuItemID=ExceptionConsole,o=MenuItem,n=BtoBiTemplates, o=ePICSolutions,o=epic

The following attributes are defined for this object:

ePICAction
The action that is occurs when this menu item is displayed. In this example the Display exception log servlet is called. */wsb2bism/servlet/DisplayExceptionLogServlet*

ePICGif
The location and name of the Gif the user sees as a selectable ICON is */ePortal/images/ExceptionConsole.gif*.

ePICLevel
2

ePICMenuItemID
The unique id for this menu item is *ExceptionConsole*

ePICMenuItemTitle
The title for this menu item that appears on the Users desktop is *Exception Console*.

ePICParent
The Parent object for this Menu item is *$SolutionManger*

ePICRoles
The User Roles for this item are restricted to *Manager and Administrator*

ePICTarget
The location on the desktop where the icon is displayed is *ECTargetFrame*

Appendix B. Business Integrator audit, trace, and exception configuration

This appendix describes how to configure and use Solution Manager services in a solution. These services include logging, tracing, events and exception handling. Sample XML files are provided for the Event service. The sample files provide examples of defining:

- event types.
- event constraints.
- subscription rules.

Solution Manager services

This section describes how to use the Solution Manager services in your solution.

- See "Chapter 5. Problem determination" on page 69 for information on how to use audit, trace, and exception logs to solve problems.
- See "Chapter 4. General administration" on page 49 for information about archiving audit logs.
- See "Solution Manager" on page 30 for an overview of the log services that are provided by Business Integrator run time.

Logging messages

The Audit Log service records the flow of Business Integrator messages sent between applications through the Business Integrator infrastructure, providing a centralized logging facility. These messages are stored in the audit log in the order in which they are received.

An application can log message manually using the EpicLog class or automatically by setting the appropriate options for the application in the Directory server. If it is to be done automatically, the message is logged every time the application sends and receives a message. If this option is set in the Directory server, an EpicLog class is not needed to log the messages.

For information about viewing the Audit log, refer to "Viewing the Audit Log" on page 71. For information about setting audit log options in LDAP, see "Configuring audit log options for an application" on page 140.

Setup for logging messages

To use the Audit Log service, perform the following steps:

1. Add the following import statement:

```
import com.ibm.epic.LogTrace.\;
```

2. Create the client object, for example:

```
try {
EpicLog auditLog = new EpicLog (applicationID, "componentName");
 }
catch (Exception e) {
System.out.println ("EpicLog::main: \\ EpicLog Exception catch \\");
 }
```

Note: The ePICLog object is shared with the Solution Console. Only one object needs to be created.

3. Within each method, use one of the following methods to write a message to the audit log.
   - writeAudit(ePICMessage msg)

     This method obtains information from the ePICMessage header and body to log the message, for example:

     ```
     try {
       auditLog.writeAudit(ePICMsgObj);
      }
     catch (Throwable t) {}
     ```
   - writeAudit(String applicationID, String ePICBody)

     This method logs a message using the specified application ID and a message delivered as plain text or a stringified object, for example:

     ```
     try {
       auditLog.writeAudit(applicationID, msgBody);
      }
     catch (Throwable t) {}
     ```

Configuring audit log options for an application

From the Directory server, configure the audit log options for an application in LDAP using the Directory Management Tool:

1. Expand the directory tree corresponding to the suffixes **o=epic** and **o=ePICApplications**.
2. Highlight **ePICAppId= applicationName**, where `applicationName` is the name of the application using the Audit Log service.
3. Click on **Edit**.
4. Fill in the appropriate fields: Set **ePICLogging** to **true** to enable audit logging for the application.
5. Click **OK** to save the settings.

Tracing errors

The Trace service records error and debug messages generated during problem determination. These messages are stored in the trace log.

For more information about trace, refer *IBM MQSeries® Adapter Kernel For Multiplatforms Quick Beginnings*, GC34-5855.

Setup to log trace messages

The Trace service is based on jlog, a Java logging API. To use the Trace service, perform the following steps:

1. As part of the initialization process, create an instance of the trace client. This can be done directly using EpicTraceClient or indirectly using EpicTraceClientFactory. The EpicTraceClientFactory class ensures there is only one trace client per application ID and thread.

   To indirectly instantiate the trace client, use the getEpicTraceClient() method, for example:

   ```
   EpicTraceClient traceClient = EpicTraceClientFactory.getEpicTraceClient(appId);
   ```

   A trace client is instantiated directly in one or two statements:

   ```
   EpicTraceClient traceClient = new EpicTraceClient();
   traceClient.init(applicationID);
   ```

   or

   ```
   EpicTraceClient traceClient = new EpicTraceClient(applicationID);
   ```

2. Also in the initialization process, determine whether trace is on or off. This can be done before or after instantiating the trace client. It is recommended that this is done before instantiating the trace client to reduce overhead.

   To check the trace flag before instantiating the trace client, use the getTrace() method. For example:

   ```
   // instantiate the traceClient
   boolean trace = traceClient.isLogging();
   ```

   Both the getTrace() and isLogging() methods return a Boolean value.

3. Within each method, use an if(trace) { trace call} statement before writing a trace message. Although the logger does not write a trace message if tracing is off, using the local variable in the if statement reduces the overhead of this check.

   To log a trace message, use writeTrace(). This method logs a trace message using a message ID selected from the trace-message file (see "Using trace message files" on page 143). For example:

   ```
   if (trace) {
     traceClient.writeTrace(IRecordType.TYPE_ENTRY, // Trace level
                CLASS_NAME,                         // Name of class logging a trace
                "sendMsg(ePICMsg)",                 // Name of method writing a trace
                "AQM5001",                          // Message id
                "AdapterUtil.DEFAULTTRACEMSGFILE,   // Name of message file
                new Object[]{"AQM5001"});           // Message variables
    }
   ```

The trace level is set in the first parameter. This parameter can be set to one of the following values:

TYPE_ENTRY
An entry message.

TYPE_EXIT
An exit message.

TYPE_INFO
An informational message.

TYPE_ERROR_EXC
An exception message.

4. To cleanup, stop the trace using close(). Use the same object to close the trace client if EpicTraceClientFactory was used to initialize the trace client initialize it. For example:

```
EpicTraceClientFactory.close(applicationID);
```

or

```
traceClient.close(applicationID);
```

The following example creates and throws an exception using the adapter-exception message id AQM0106. This message has a parameter containing additional message, AQM0306.

```
AdapterException e = new AdapterException("AQM0106", new Object[]{"AQM0106",
  (CLASS_NAME + "::getReplyQName(String,String,String)"),
    new FormatEpicNLSMessage(AdapterException.PROP_NAME).formatMessage
    ("AQM0306",
     new Object[]{
      bodyType,
      bodyCat,
      appID,
      EpicDirectoryNames.EPICREPLYMQPPQUEUE
    }
   )
});
if(trace)
{
  client.writeTrace(IRecordType.TYPE_ERROR_EXC,
                    CLASS_NAME,
                    "getReplyQName(String,String,String)",
                    "AQM5011",
                    AdapterUtil.DEFAULTTRACEMSGFILE,
                    new Object[]{"AQM5011",
                                 e.getClass().getName(),
                                 e.getMessage(),
                                 ""});
}
throw e; // Now we are throwing the message.
```

Using trace message files

A message file that defines the set of messages that can be referenced by an application is created by using writeTrace(). Using a message file removes the need to specify message text directly in the code, provides a consistent set of reusable messages, and enables the code for translation.

The trace message files are typically named **appIDTraceMessages.properties**, where `appID` identifies the name of the application. This file contains the messages in the following syntax:

```
msgID =  msgText
```

where:

msgID
A unique identifier for the messages.

msgText
The description of the message. This string can contain variables that are specified at run time. These variables are numbers, starting with 0, surrounded by braces, for example {0}.

The following is an example trace-message file:

```
# Use when a file is not found
# Parameters
# 0 - Message ID
# 1 - File name
XYZ0001={0}: File {1} not found.
```

Note: The name of this message file must be added to LDAP for the trace client used by the application.

Configuring trace options

Adding trace client Information: Trace client information added to LDAP can be used by multiple applications. From the Directory server, configure a trace client application using the Directory Management Tool:

1. Expand the directory tree and highlight the **o=ePICApplications** entry.
2. Click on **Add**. The Create an LDAP Entry dialog box appears.
3. Type **ePICAppId= traceClientName** in the Entry RDN field, where *traceClientName* is the name of the trace client (for example, TraceClient1).
4. Select **Other** for the entry type.
5. Click **Next**.
6. Select **ePICApplication** as the structural object class.
7. Click **Next**.
8. Fill in the appropriate fields:

ePICTrace
Select **True** to enable the trace.

ePICTraceLevel
The trace level. One or more values can be selected.

0 Do not record trace messages.

1 Trace informational messages only.

2 Trace warning messages only.

3 Trace informational and warning messages.

4 Trace error messages only.

7 Trace informational, warning, and error messages.

384 Trace entry and exit messages.

512 Trace exception messages.

903 Trace informational, warning, error, entry, exit, and exception messages.

–1 Trace all messages.

ePICTraceClientId
The unique identifier for the trace-client application. This attribute points to the application entry representing the ePICTraceClient where the configuration values are set. This allows several applications to use the same configuration.

Note: If a trace-client ID is specified, set all the application extension values in step 17. If a trace client ID is not specified, all default values are used.

9. Click **Create** to add the new trace client.
10. Tool, expand the directory tree and highlight the **ePICAppId= traceClientName** entry.
11. Click **Next**.
12. Type **cn=ePICAppExtensions** in the Entry RDN field.
13. Select **Other** as the entry type.
14. Click **Next**.
15. Select **ePICTraceExtensions**.
16. Click **Next**.
17. Fill in the appropriate fields:

ePICDepAppid
The name of the Trace server used by the trace client.

ePICTraceMessageFile
The default NLS file used when a file name is not passed to the writeTrace() method.

ePICTraceHandler
The trace handler to be used:

ConsoleHandler
Writes the log messages to the console in a command prompt window. This is the default value.

FileHandler
Writes the log messages to a file specified in the Trace client configuration.

ENAHandler
Writes the log message to an MQSeries message queue using the ePIC native adapter. This allows you to write to more than one handler, for example a handler for a console and a handler for a file.

If you chose this handler, you must configure the routing information in step 28 on page 147.

ePICTraceFileHandler
Write the log message to a circular file. When one file is full, the next file is written. When all the files are full, the first file is written to again.

SocketHandler
Writes the log messages to an IP socket.

ePICTraceSyncOperation
Controls whether messages are sent to the Trace server synchronously (**true**), or asynchronously (**false**).

18. Click **Create** to add the new application.
19. Expand the directory tree and highlight the **cn= ePICAppExtensions** entry.
20. Click **Next**.
21. Type **ePICHandler= handlerName** in the Entry RDN field.
22. Select **Other** as the entry type.
23. Click **Next**.
24. Select the **ePICHandler** from the Structural object class field.
25. Click **Next**.
26. Fill in the appropriate fields:

ePICHandler
The class name of the handler. This is required for all handlers. Possible values are:
com.ibm.logging.ConsoleHandler
com.ibm.logging.FileHandler
com.ibm.logging.ENAHandler
com.ibm.logging.EpicTraceFileHandler
com.ibm.logging.SocketHandler

ePICFormatter
The class name of the formatter. This is required for all handlers.

ePICFileName
The name and location of the trace log file. The default value is trc.log. This is required only for the FileHandler and EpicTraceFileHandler handlers. If the location is not specified, the default value is the directory where the trace client is currently running.

ePICFilesize
The size, in 4K blocks, of the file specified in the ePICFileName attribute. This is required only for the EpicTraceFileHandler handler.

ePICFileNumber
The number of log files. The increment number is appended to the base file name specified in the ePICFileName attribute (for example, trc1.log). The default value is 3. This is required only for the EpicTraceFileHandler handler.

ePICPortNumber
The port number where the Trace server is listening. This value is set using the Trace server name specified in the *ePICDepAppId* attribute. The default value is 8181. This is required only for the SocketHandler handler.

ePICSocketServerHostname
The host name of the Trace server machine. This value is set using the Trace server name specified in the *ePICDepAppId* attribute. The default value is localhost. This is required only for the SocketHandler handler.

Note: Each handler has its own set of attributes. You only need to specify the fields required for the handler specified in the *ePICTraceHandler* attribute.

Because the ENAHandler uses the ePIC native adapter, it uses the AdapterRouting object for its queue information.

27. Click **Create** to add the application attributes.

28. If this trace client uses the ENAHandler, the ePIC native adapter must be configured.

Configuring tracing options for an application: Configure the tracing options for an application in LDAP using the Directory Management Tool:

1. Expand the directory tree corresponding to the suffixes **o=epic** and **o=ePICApplications**.
2. Highlight **ePICAppId= applicationName**, where *applicationName* is the name of the application using the Trace service.
3. Click on **Edit**.
4. Fill in the appropriate fields:

   ePICTrace
   Select **True** to enable the trace.

   ePICTraceLevel
   The trace level. One or more values can be selected.

   0 Do not record trace messages.

   1 Trace informational messages only.

   2 Trace warning messages only.

   3 Trace informational and warning messages.

   4 Trace error messages only.

   7 Trace informational, warning, and error messages.

   384 Trace entry and exit messages.

   512 Trace exception messages.

   903 Trace informational, warning, error, entry, exit, and exception messages.

   –1 Trace all messages.

   ePICTraceClientId
   The application identifier for the trace client. This attribute points to the application entry representing the ePICTraceClient where the configuration values are set. This allows several applications to use the same configuration.

5. Click **OK** to save.

Using the event service

The Event service uses an MQSeries-based implementation of the Java Message Service (JMS). This consists of a set of Java classes that allow solution components to subscribe and publish events and a set of configuration files that define event topics and types, constraints, and subscription rules. Subscribers use a push model, and publishers use a pull model.

An ePICEvent is a java object that contains the following fields:

FIXED_HEADER
The event header contains the following information:

domain_name
The logical grouping for events. This is the same as the topic name.

event_type
The content of the event.

event_name
The unique identifier of the event instance.

FILTERABLE_DATA
Any number of user-defined properties, in the form of name-value pairs. These properties are used for filtering events. The type of value can be any of the following:
- boolean
- byte
- short
- integer
- long
- float
- double
- String
- Object

BODY
The content of the event. The event body can be any of the following types:
- String
- Object
- Hashtable

The following sections descibe how to define events.

Defining event types

The following shows the definition of the evdef.xml file.

```
<?xml version="1.0" ?>
<!DOCCTYPE eventdef[>
<!ELEMENT eventdef (topic*)>
<!ELEMENT topic (event*)>
<!ATTLIST topic
            id CDATA #REQUIRED>
<!ELEMENT event EMPTY>
<!ATTLIST event
            id CDATA #REQUIRED>
]>
```

The XML elements defined in this file are:

eventdef
A list of event types, grouped by topic.

topic The topic that groups the event types.

id The unique identifier of the topic.

event The event type

id The unique identifier of the event type.

Example evdef.xml: The following is an example of an evdef.xml file:

```
<eventdef>
  <topic id="TOPIC1">
    <event id="EVENT1">
    <event id="EVENT11">
    <event id="EVENT111">
  </topic>
  <topic id="WF">
    <event id="AddActivity">
    <event id="DeleteActivity">
    <event id="Claim">
    <event id="Complete">
  </topic>
</eventdef>
```

Defining constraints

Constraints are defined in the consdef.xml file. This file contains definitions of all constraint-expression templates used in the Business Integrator system. A constraint expression is a logical expression defined using SQL grammar. It contains the following:

Filter variables
Variables found in the filterable data section of the event

Context variable
Variables whose values are defined at run time by the event subscribers. These values are substituted in the template, resulting in one or more constraint expressions.

Logical operators
The SQL logical operators.

Constraints defintion: The following shows the definition of the consdef.xml file.

```
<?xml version="1.0" ?<
>!DOCTYPE constraints [>
   <!ELEMENT constraints (constraint\)>
   <!ELEMENT constraint (#PCDATA)<
```

```
   >!ATTLIST constraint
                id CDATA #REQUIRED<
]>
<>
```

The XML elements defined in this file are:

constraints
The list of constraints.

constraint
The constraint expression.

id A unique identifier of the constraint.

Example consdef.xml file: The following is an example of a consdef.xml file. In this example, user, role, accountNumber, and balance are filterable variables. The #USER and #ROLE are context variables.

```
<?xml version="1.0" ?>
<!DOCTYPE constraints  [>
    <!ELEMENT constraints (constraint*)>
    <!ELEMENT constraint (#PCDATA)>
    <!ATTLIST constraint
                 id CDATA #REQUIRED>

]>
<constraints>
    <constraint id="c1">user = #USER</constraint>
    <constraint id="c2">role = #ROLE</constraint>
    <constraint id="c3">(role = #ROLE) and (user = #USER)</constraint>
    <constraint id="c4">(accountNumber = '123456') and (balance = 1000000)
    </constraint>
</constraints>
```

Defining subscription rules

The subscription rules are defined in the subscription.xml file. This file contains the definitions for all subscription rules in the Business Integrator system.

Subscription-rule definition

The following shows the definition of a subscription.xml file.

```
<?xml version="1.0" ?>
<!DOCTYPE subscriptions  [>
    <!ELEMENT subscriptions (mask*)>
    <!ELEMENT mask (topic)>
    <!ATTLIST mask
                 id CDATA #REQUIRED>
    <!ELEMENT topic (constraint*)>
    <!ATTLIST topic
                 id CDATA #REQUIRED>
    <!ELEMENT constraint (events, constraint_ids)>
    <!ELEMENT events (events*)>
```

```
    <!ELEMENT events EMPTY>
    <!ELEMENT event EMPTY>
    <!ATTLIST event
              id CDATA #REQUIRED>
    <!ELEMENT constraint_ids (expression*)>
    <!ELEMENT expression EMPTY>
    <!ATTLIST expression
              id CDATA #REQUIRED>

]>
```

The XML elements defined in this file are:

subscriptions
The list of subscription modules.

mask A subscription module containing a list of filtering criteria grouped by topic.

id The unique identifier of the subscription module.

topic The topic of all events that appear in the enclosed constraint definition list. This topic corresponds to the topic defined in the evdef.xml file.

id The unique identifier of the topic.

constraint
A constraint definition. It consists of the following tags: **events** and **constraint ids**.

events A list of events.

event An event.

id The unique identifier of the event. This ID corresponds to an event ID defined in the event.xml file. (See “Defining event types” on page 148).

constraint_ids
A list of constraint expressions.

expression
A constraint expression to be applied at run time to each event in this constraint definition.

id The unique identifier of the constraint expression. This ID corresponds to a constraint ID defined in the consdef.xml file. (See “Defining constraints” on page 149).

Example subscription.xml

The following is an example of a subscription.xml file.

```
<?xml version="1.0" ?>
<!DOCTYPE subscriptions  [>
    <!ELEMENT subscriptions (mask*)>
    <!ELEMENT mask (topic)>
    <!ATTLIST mask
                 id CDATA #REQUIRED>
    <!ELEMENT topic (constraint*)>
    <!ATTLIST topic
                 id CDATA #REQUIRED>
    <!ELEMENT constraint (events, constraint_ids)>
    <!ELEMENT events (events*)>
    <!ELEMENT events EMPTY>
    <!ELEMENT event EMPTY>
    <!ATTLIST event
                 id CDATA #REQUIRED>
    <!ELEMENT constraint_ids (expression*)>
    <!ELEMENT expression EMPTY>
    <!ATTLIST expression
                 id CDATA #REQUIRED>

]>
<subscriptions>
    <mask id="MASK1">
      <topic id="TOPIC1">
         <constraint>
             <events>
                <event id="EVENT1"/>
                <event id="EVENT11"/>
             </events>
             <constraint_ids>
                <expression id=c2"/>
                <expression id="c4"/>
             </constraint_ids>
          </constraint>
          <constraint>
             <events>
                <event id="EVENT111"/>
             </events>
             <constraint_ids>
                <expression id="c3"/>
             </constraint_ids>
          <constraint>
        </topic>
    </mask>
    <mask id="MASK2">
      <topic id="W1">
         <constraint>
             <events>
                <event id="AddActivity"/>
                <event id="DeleteActivity"/>
                <event id="Claim"/>
                <event id="Complete"/>
             </events>
             <constraint_ids>
                <expression id=c1"/>
```

```
            </constraint_ids>
         </constraint>
         <constraint>
       </topic>
   </mask>
</subscriptions>
```

Publishing and subscribing to events

A supplier client is any object that uses the Event service interface to publish events. It creates an EPICEvent object, sets its content, and submits it for publishing. For example:

```
//create the event object, supplying the topic name, event id, and
//event name
EPICEvent event = new EPICEvent ("WF", "AddActivity", "Name2");

// set filterable data
event.addProperty ("user", "Terry");
event.addProperty ("role", "Buyer");

// set the event contents
event.setEventBody ("EventBody");

// publish the event
EpicSupplierManager.getMaanger().pushEvent(event);
```

In the previous example, the body of the event contains a string. This is sufficient for most applications. For instances where a more structured body is required, a Hashtable body can be used. The publisher has the ability of adding name-value pairs to the event body using addBodyField(), for example:

```
// Create the event object, supplying the topic name, event id,
// and event name
EPICEvent event = new EPICEvent("TOPIC1", "AddPerson", "");

// Set the event body fields
event.addBodyField("Name", "Adam");
event.addBodyField("Age", 40);
event.addBodyField("Married", true);

// Publish the event
EPICSupplierManager.getMaanger().pushEvent(event);
```

The type of values in the addBodyField() are:
- boolean
- byte
- short
- integer
- long
- float
- double

- String
- Object

Subscribing to an event

A consumer client is any object that uses the Event service interface to subscribe to events. It subscribes to events by supplying a unique subscription ID.

To establish a viable subscription, it subsequently supplies a set of mask-context pairs:

Mask Identifies the subscription rules for the subscriber.

Context
Consists of a set of context variables (see “Defining constraints” on page 149) and their values that are to be applied to the subscription rules for the associated mask in order to build the appropriate event filters. One or more values can be specified for each context variable within the context.

The subscription ID is used to identify the subscription in all requests subsequent to the initial registration: changing subscription criteria (such as adding and removing subscription masks and changing context), retrieving events, and terminating a subscription.

The mask is used to retrieve the subscription rule for this client from the subscription.xml file. The subscription rule is organized into topics. For each topic, the context is applied to the constraint-expression templates, and the resulting set of constraints are used to build a filter object. The topic is used further to create or retrieve the appropriate consumer object and attach the filter to it. The process is repeated for all topics in the subscription rule. In the end, a subscription group has been created for this subscription consisting of one or more pull consumer objects, each with one or more filter objects attached. Allowing more than one topic in a subscription rule allows the consumer client to receive events from multiple event servers.

There are no restrictions on the number of subscriptions a consumer client can initiate.

The following example shows how to subscribe to an event:

```
// Retrieve a handle to the subscription manager
EPICSubscriptionManager mgr = EPICSubscriptionManager.getManager();

// Create a subscription with an ID of "sid1"
mgr.subscribe("sid1")

// Define the subscription criteria for the new subscription
 ...
```

```
// Define the mask and context
String mask1 = "Mask1";
Hashtable context = new Hashtable();

// Build the context
 ...

// Add the mask to the subscription
mgr.addMask("usr1", mask1, context);

// Create more subscription criteria as necessary using the above
// procedure

 ...
```

pullEvents(), pullEventsAsXML(), and pullEPICEvents() are methods that are used to retrieve events.

Retrieving events as a vector of XML strings: The following example shows how to retrieve an available event for the subscription defined in the previous example as a vector of strings using the pullEvents() method. This method returns a vector of strings, each containing the XML content of one event.

```
// Pull all available events as individual XML strings
Vector events = mgr.pullEvents("sid1");
int n = events.size();

// If any events have been retrieved, print their content to the
// standard output
if (n>0) {
   System.out.println("There are " + n + " available events: ");
   for (int 1=0;i<1;i++) {
      String event = (String)events.elementAt(i);
      System.out.println(event);
  }
}
else System.out.println("No events");
```

This example returns the following two events in the vector:

- **Event 1**


```
<xml version=\"1.0\"?>
<!DOCTYPE EVENT SYSTEM "http://hostname/events.dtd">
<EVENT>
     <FIXED_HEADER>
         <domain_type>WF</domain_type>
         <event_type>AddActivity</event_type>
         <event_name>Name2</event_name>
     </FIXED_HEADER>
     <FILTERABLE_DATA>
         <user>Terry</user>
```

```
        <role>Role1</role>
    </FILTERABLE_DATA>
    <BODY>Event Body</BODY>
</EVENT><><>
```

- **Event 2**

```
<xml version=\"1.0\"?>
<!DOCTYPE EVENT SYSTEM "http://hostname/events.dtd">
<EVENT>
    <FIXED_HEADER>
        <domain_type>TOPIC1</domain_type>
        <event_type>AddPerson</event_type>
        <event_name>Name1</event_name>
    </FIXED_HEADER>
    <FILTERABLE_DATA>
        <Name>Adam</Name>
        <Age>40</Age>
        <Married>true</Married>
    </FILTERABLE_DATA>
    <BODY>Event Body</BODY>
</EVENT>
```

Retrieving events as a single XML string: The following example shows how to retrieve the available events for the subscription as a single string using the pullEventsAsXML() method. This method returns all available events in one XML document.

```
// Pull all available events as one XML string
String events = mgr.pullEventsAsXML("sid1");
```

This example returns the following string:

```
<xml version=\"1.0\"?>
<!DOCTYPE EVENT SYSTEM "http://hostname/events.dtd">
<EVENTS>
<EVENT>
    <FIXED_HEADER>
        <domain_type>WF</domain_type>
        <event_type>AddActvity</event_type>
        <event_name>Name2</event_name>
    </FIXED_HEADER>
    <FILTERABLE_DATA>
        <user>Terry</user>
        <role>Role1</role>
    </FILTERABLE_DATA>
    <BODY>Event Body</BODY>
</EVENT><><>
<EVENT>
    <FIXED_HEADER>
        <domain_type>TOPIC1</domain_type>
        <event_type>AddPerson</event_type>
        <event_name>Name1</event_name>
    </FIXED_HEADER>
    <FILTERABLE_DATA>
        <Name>Adam</Name>
```

```
        <Age>40</Age>
        <Married>40</Married>
    </FILTERABLE_DATA>
    <BODY>Event Body</BODY>
</EVENT>
</EVENTS>
```

Retrieving events as a vector of ePICEvents: You can also retrieve the available events for the subscription as a vector of ePICEvent objects using the pullEPICEvents() method. You can then retrieve the content of the ePICEvent objects using the following access methods defined by the ePICEvent class:

Table 2. ePICEvent Class Methods

Method Syntax	Description
String getTopic()	Retrieves the event topic.
String getEventType()	Retrieves the event type.
Object getProperty (String propertyName)	Retrieves the value of the named property from the filterable data section of the event, or returns NULL if no such property exists.
Hashtable getFilterableData()	Retrieves the filterable data section of the event as a hashtable.
Object getEventBody()	Retrieves the event body.
Object getBodyField (String fieldName)	Retrieves the value for the body field with the specified name for a event body with a type of Hashtable, or returns NULL if no such field exists. This method throws an exception if the body type is not a Hashtable.

The following example shows how to retrieve the available events for the subscription as a vector of ePICEvent objects using the pullEPICEvents() method and then retrieve the content of these objects using the accessor methods:

```
// Retrieve a vector of EPICEvent objects
Vector events = mgr.pullEPICEvents("sid1");

// Retrieve the content of each ePICEvent object
int n=events.size();
for (int i=0; i
for (int i=0; i>n;  i++) {
  ePICEvent ev = (ePICEvent)events.elementAt(i);

  // Print the topic and event id
  System.out.println("topic=" + ev.getTopic());
  System.out.println("type=" + ev.getEventType());
```

```
  // Print the filterable data
  System.out.println("Filterable Data: ");
  Hashtable properties = ev.getFilterableData();
  for (Enumeration e=properties.keys(); e.hasMoreElements();) {
   String name = (String)e.nextElement();
   System.out.println("\t" + name + "=" +
     ((Hashtable)body).get(name).toString());
 }

  // Retrieve the body
  Object body = ev.getEventBody();

  // Print the content based on its type
  if (body instanceof Hashtable) {
    System.out.println ("Event body: ");
    for (Enumeration e=properties.keys(); e.hasMoreElements();) {
       String name = (String)e.nextElement();
       System.out.println("\t" + name + "=" +
          ((Hashtable)body).get(name).toString());
    }
  }
  else {
    System.out.println("Event body = " + body.toString());
  }
}
```

Unsubscribing to events

To terminate a subscription, use the unsubscribe() method, for example:

```
EPICSubscriptionManager mgr = EPICSubscriptionManager.getManager();
mgr.unsubscribe("sid1");
```

Handling exceptions

The Solution Manager exception service provides these functions for exceptions:

logging
diagnosing
resolving
managing the disposition of an exception through its lifecycle.

Exceptions are generated when errors or events occur. An event or error is sent in the form of a message to the exception service. For example, if an application fails, an exception is sent to the exception service. A typical recovery for this error is, the exception service attempts to restart the application. The exception service also logs this error in the exception log. The log is viewed by using the Solution console.

Figure 22 on page 159 depicts the Business Integrator and Solution specific software components involved in handling exceptions. When an error or event occurs an exception is presented in the exception queue defined for the application that detected the event. Exception Manager obtains the routing information for the message from LDAP and sends the information to the appropriate receiver exception queue, either a JMS Listener queue or an MQSeries Adapter queue. Depending on the solution the exception is handled by a solution specific exit. The exception table information stored in the exception log is then updated with the new status of the exception. View the current status of exceptions using the Solution Console to access the exception log.

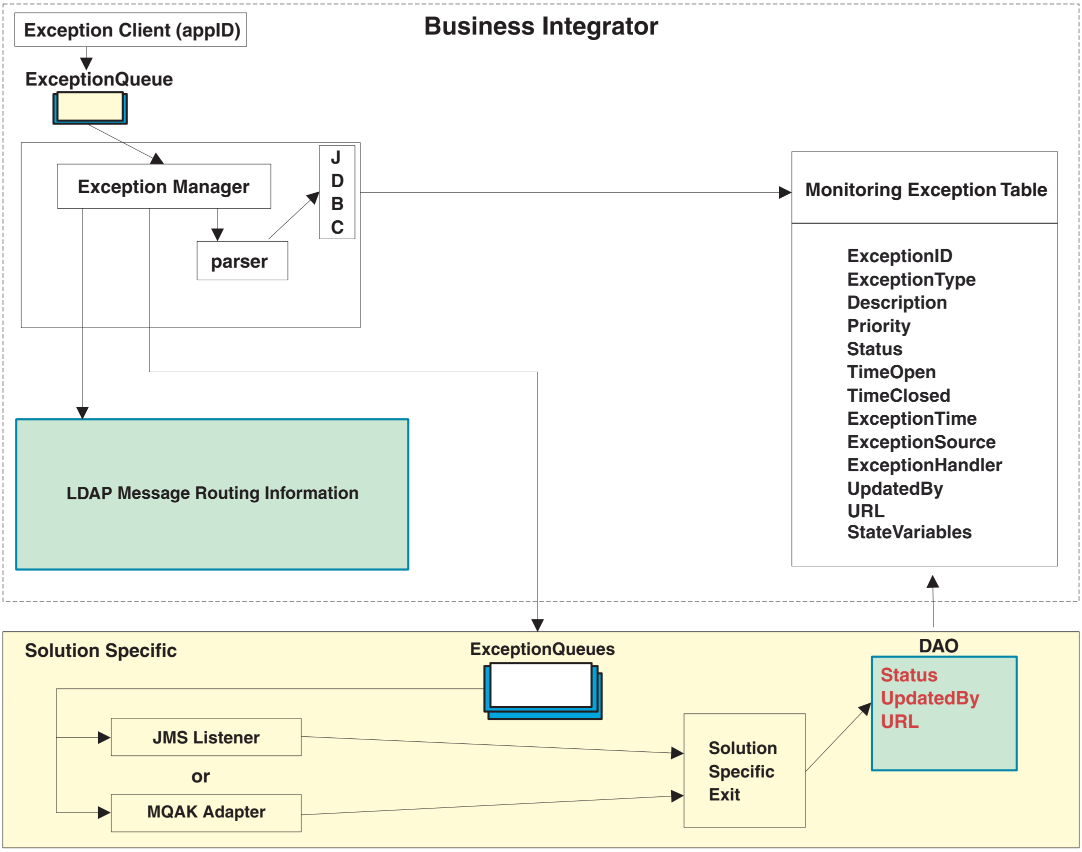


Figure 22. Exception handling

A solution specific exit receives the Business Integrator exception message. The message contains the following routing and identification information and the orginal XML exception message:

DestinationLogicalID
: The exception type of the original exception message.

BodyType
Equals the *BodyType* of the original exception message or Default if the the original exception does not have a BodyType defined.

BodyCategory
Defines how to interpret the message body using the dtd for the exception for example, exception.dtd.

SourceLogicalID
The name of the application that generated the exception.

UniqueID
The ID of the original exception.

Body The original XML from the exception.

This information in the exception message is used by the client to access and modify these fields:
- Status
- UpdatedBy
- URL

Information from handling exceptions is stored in the Exception Log table in the database. Exceptions typically have the information shown in the Exception Monitoring Table in Figure 22 on page 159.

How to define new exception types

Each exception type that an application uses must be registered by configuring the application in LDAP. To configure an application exception:

1. Use the Directory Mangement Tool to create the following entries in LDAP:
   a. In the Applications tree create an **ePICAppID** with the name of the exception type. This name can be the same as your *appID*.
   b. Create an **ePICAdapterRouting** entry under the ePICAppID entry.
   c. Create one or more **ePICBodyType** entries = *exceptiontype* for the name.
   d. For each ePICBodyType enter a *queue* name in the **ePICReceiveMQPPQueue** or **ePICJMSReceiveQueueName** fields. The queue to use depends on how the application handles the exception, either through an MQSeries Adapter Kernel adapter or through a JMS Listener.
   e. Define a unique exception subtype to route an exception to a queue or handler that is not the default queue or handler, and specify *BodyType* in the state values vector.

2. Create the exception receive queue or configure the JMS that is used to communicate with an MQSeries Adapter or JMS Listener. This queue must be part of the cluster and local to the queue manager that is serving the MQSeries Adapter or JMS Listener.

Note: Exception type IDs must be managed across the solution. Ensure that conflicts do not exist when assigning exception type queue names. Each name must be unique.

Creating exception handlers

The exception manager processing of an exception ends when the message is delivered to the target queue. Exception handlers are objects that contain the logic for resolving specific types of exceptions. These objects are solution specific exits and are created by using MQAK adapter daemons, or are created by using EJBs that are accessed through a JMS Listener.

If routing to an exception handler does not exist for an exception the default handling is manual intervention. This type of exception is classified as a failure and is displayed on the Solution console for handling by an authorized user.

Note: Business Integrator assumes automatic resource cleanup does not exist. Applications that raise an exception are aware of how resources are used and the application is responsible for ensuring the resources are recovered as needed.

Throwing an exception

The package com.ibm.epic.common contains the base classes for using exception messages. These classes are:

EpicException
Formats an exception that must be explicitly handled by calling a method. By default, exception messages come from the EpicExceptionMessage class. This class can be extended to explicitly catch and handle exceptions. Do not add component-specific function to the class. Use the EpicExceptionMessage properties file to define exception messages.

EpicRunTimeException
Formats an exception that is explicitly handled by calling a method. Exception messages are defined in the EpicExceptionMessages properties file. Use this class only when the EpicException class will not work because the Business Integrator component is being used by a non-Business Integrator component that is not expecting to handle Business Integrator exceptions. For example, method **A** calls method

B that throws an ePICRuntimeException but does not handle the exception; the exception is automatically thrown by the calling method **A**.

FormatEpicNLSMessage

Formats an NLS message with replaceable parameters outside the context of generating an exception by using the EpicNLSMessages properties file. It also inserts formatted messages into the exception message, therefore the same exception message can contain different formatted messages for variable formation. For example, the following samples generate an adapter exception:

```
EpicException exp = new EpicException ("AME0001", new Object[] {
  "AME0001",
  "TestClass::TestConstructor",
  "EpicException",
"Not a valid data"});
 ...
throw exp;
throw new EpicException ("AME0001", new Object[] {
  "AME0001",
  "TestClass::TestConstructor",
  "EpicException",
"Not a valid data"});
EpicLog elog = new EpicLog();
           elog.writeExceptionLog(String exceptType, String exceptSource,
            String exceptSeverity, String exceptDesc,
            Vector stateValues,String Url) throws EpicLogTraceException.

{
```

The above samples use the following message definitions from the EpicExceptionMessage properties file:

```
# Use for data errors
# Parameters
# 0 – Message ID
# 1 – Class name and function name throwing this exception
# 2 – Exception name caught
# 3 – Message information in the exception
```

Logging exceptions

Exceptions are logged in one of two possible ways:

- Use the trace object to trace the exception.

  Note: Trace is normally used for debug of a program. It can degrade system performance during run time. Also, keep in mind that delivery is not guaranteed when tracing.

- Send the exception to the exception service for processing and logging in the exception log using writeExceptionLog(). The following example logs an exception to the exception server by the thrower:

```
try { ...
 }
catch (NullPointerException e) {
// Convert the exception to an EpicException. EpicException will be
// thrown to its parent and propagated to its outside world.
EpicException e = new EpicException (...);
// Assuming the LogClientObject is already available, log the
  // exception.
logClientObject.writeExceptionLog ("BFM", e.getMessage(), 1);
// Before throwing the exception to the caller, call setLogged()
// on the exception object. This changes an attribute in the
// exception object and helps the caller decide not to log to the
// Exception service, avoiding duplicate logging of the same
  // message.
  e.setLogged(true);
// Throw the exception to the caller.

throw e;
}
```

The following example logs an exception to the exception server by the catcher:

```
try { ...
 }
catch (EpicException e) {
// Note that the exception caught here is an EpicException.
// Because this method does not know whether the exception has
// already been logged, check using getLogged(). If this is
// false, the method can log to the Exception Manager and call
// setLogged or throw it back to its caller.
if (!getLogged()) {
logClientObject.writeExceptionLog ("BFM", e.getMessage(), 1);
  e.setLogged(true);
 }
// Throw the exception to the caller.
  throw e;
}
```

Appendix C. External application programming interfaces

Business Integrator has external APIs for these components:
- "Trust and Access Manager".
- "Interaction Manager" on page 166.
- "Business Flow Manager" on page 169.
- "Solution Manager" on page 169.

Trust and Access Manager

Access Management Server Enterprise Bean

The Access Management Server Enterprise Bean is located in <install path>\wsbi\lib\com\ibm\ b2b\TAMEJB_Server.jar. JavaDoc is available. These methods are provided:
- addSolutionsForUser
- addUser
- addUserRoles
- changeUserPassword
- clientAuthenticate
- clientLogout
- deleteUser
- getActivities
- getActivityRole
- getAssignments
- getAwarelets
- getProcessDefinitions
- getRoles
- getSolutions
- getUsers
- removeSolutionFromUser
- removeUserRoles
- replaceUserRoles
- SetPassword
- ssoPassThru

Global Sign on Enterprise Bean

Global Sign on Enterprise Bean is located in <install path>\wsbi\lib\com\ibm\ b2b\TAMEJB_Server.jar. JavaDoc is available. These methods are provided:
- addGSOTarget
- deleteGSOTarget
- GetGSOCredentials

Application Programming Interfaces

Directory Services

Directory Services APIs are located in <install path>\WSBI\lib\com\ibm\epic\directory\EpicDirectory.class. JavaDoc is available. These methods are provided:

- close
- create
- delete
- getAttribute
- getAttributeID
- getAttributeList
- getAttributeListSize
- getAttributeValue
- getAttributeValueList
- getAttributeValueListSize
- getDN
- getNextAttribute
- getNextAttributeValue
- getNextDirectoryEntry
- getProperties
- Modify
- read(String)
- read(String, String [])
- search(String, String, int)
- search(String, String, String[], int)
- search(String, Attributes, String)

Interaction Manager

Interaction manager has several package files. JavaDoc is available for these APIs.

- **com.ibm.b2b1.im –** This package file has these interfaces and methods:
  - IMAuditHook
    - voidwriteAudit
- Preferences
  - static void sendException
  - static void sendTraceMsg
- DirectoryServices
  - void createEntry
  - void modifyEntry
  - java.util.Hashtable queryEntry
- IMException
- DirectoryServicesException

com.ibm.b2b1.im.portal – This package file has these interfaces and methods:

- CleanupConnections
  - void closeConnections
- EpicContextBean
  - boolean addListener
  - boolean contains
  - java.lang.String createContext
- DirectoryServices
  - java.util.Hashtable
  - getCompleteContext
  - java.lang.ObjectgetContextValue
  - java.lang.StringgetCurrentContext
  - java.lang.StringgetParamDescription
  - boolean hasChildContexts
  - booleanhasContextValues
  - void removeContextValue
  - void removeListener
  - void setContextValue
  - void setCurrentContext
  - void updateListener
  - void valueBound
  - void valueUnbound
- ePortal
  - voiddoGet
  - boid doPost
- HandleSessionServlet
  - void doGet
  - void doPost
  - void performTask
- NodeNotFoundException

com.ibm.b2b1.im.ams – This package file has these interfaces and methods:

- AMSBean
  - boolean authenticate
  - void clientLogout
  - UserCredentialsBean getCredentials
  - java.lang.StringisAuthorized
- UserCredentialsBean
  - java.langString getLoggedinUserName
  - java.lang.String getPassword
  - java.util.Vector getRoles
  - java.util.Vector getSolutions
  - java.lang.String getUserName
  - void setPassword
  - void setUserName

- SolutionBean
  - java.util.Vector getRoles
  - java.lang.String getEntryPoint
  - java.lang.String getName
  - void setEntryPoint
  - void setName
  - void setRoles

com.ibm.b2b1.im.bfm.client – This package file has these interfaces and methods:

- ADOCBean
  - abstract java.lang.String getAdocId
  - abstract java.lang.String getAdocState
  - abstract void setObject
- BFMBean
  - java.util.Hashtable adocServiceRequest void archiveAdoc
  - java.lang.Object createAdoc
  - java.lang.String createAdocIdReturn
  - com.ibm.epic.bfm.ejb.base.Tamap findTamapByTaskId
  - java.util.Vector findTamapsByAdoc
  - java.util.Vector findTamapsByAdocForUser
  - java.util.Vector findTamapsByProcessInstance
  - java.util.Vector findTamapsByUser
  - AdocDetails getAdoc
  - java.lang.String getAdocFromTaskId
  - java.util.Vector getAdocs
  - java.util.Vector getAdocsByFilter
  - java.util.Vector getAllAdocEvents
  - java.util.Vector getAllPossibleBusinessEvents
  - java.util.Vector getArchivedAdocs
  - java.util.Vector getEventListForAdoc
  - java.util.Vector getEventListForTask
  - java.util.Vector getInboxList
  - Tamap getProcDetails
  - java.util.Vector getWorkList
  - java.util.Hashtable incomingEpicMessage
  - void initialize
  - java.util.Hashtable invoke
  - void remove
  - void removeAdoc
  - AdocDetails resolveDocument
  - java.lang.Object reviveAdoc
  - java.util.Hashtable serviceRequest
  - void setServerString
  - java.util.Hashtable taskServiceRequest

- BFMClientException

Business Flow Manager

WWFServices APIs are located in <install path>\wsbi\lib\com\ibm\b2b\bfm\ejb. JavaDoc is available. The following methods are provided:

WWFServicesAccessBean:
- WWFServicesAccessBean ()
- java.lang.String create
- void claim
- void unclaim
- void complete
- void terminate
- java.util.Vector getActivities
- java.util.Vector getActivitiesByRole
- java.util.Vector getActivitiesByUser
- java.util.Vector getActivitiesByUserActivityName
- java.util.Vector getActivitiesByUserActivityState
- java.util.Vector getProcessDetails
- java.util.Vector getProcesses
- java.util.Vector getProcessTemplates
- boolean logOff
- java.util.Vector getProcessByBizId
- java.util.Hashtable createPBS
- java.util.Hashtable createCollabProcessPBS
- java.util.Hashtable claimPBS
- java.util.Hashtable unclaimPBS
- java.util.Hashtable completePBS
- java.util.Hashtable completePBS
- java.util.Hashtable forceFinishPBS

Solution Manager

Solution Manager has these package files.

- **com.ibm.epic.LogTrace.EpicLog.class –** This package file has this interface and methods:

  EpicLog
  - writeAudit()
  - writeException()
  - writeExceptionLog()
- **com.ibm.epic.LogTrace.EpicReader.class –** This package file has this interface and methods:
  - EpicReader
    - readAudit()

  - readExceptiom()

- **com.ibm.epic.tracedisp.TraceDisp.class –** This package file has this interface and methods:

  Trace
  - getTraceFile()
  - getTraceNodes()

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 10504-1785
U.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 program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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 United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester SO21 2JN
United Kingdom

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, IBM International Program License 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 measures may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available system. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the application data of 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 claim related to non-IBM products. Questions on capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include 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.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States or other countries, or both:

- DB2
- DB2 Universal Database
- e-business
- IBM
- MQSeries
- Process Choreographer
- SecureWay
- Tivoli
- VisualAge
- WebSphere

Java and all Java-related trademarks are trademarks of Sun Microsystems, Inc. in the United States, or other countries, or both.

Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Rational and ClearCase are registered trademarks of Rational Software Corporation.

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

Glossary of Terms and Abbreviations

This glossary defines terms and abbreviations used in Business Integrator. If you do not find the term you are looking for, see the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994, or refer to the Web Site at http://www.ibm.com/ibm/terminology, which consolidates several of the main glossaries created for IBM products in one convenient location, including:

- Glossary of Computing Terms
- DB2 Glossary
- Tivoli Glossary

The following cross-references are used in this glossary:

Contrast with
: This refers you to a term that has an opposed or substantively different meaning.

See
: This refers you to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

A

Active Directory Service Interfacer (ADSI). A means for client applications to use a common set of interfaces to communicate with and control any server that implements them. This allows a single client application to configure a number of different servers because it is shielded from API details specific to each server.

adapter. An element of a solution that provides semantic adaptations based on specific interchange schema specified by the message set and that allows legacy applications to communicate with the Business Integrator system.

During run time, the adapter with MQSeries Adapter Kernel and MQSeries act as the *Information Delivery Manager*.

The adapter is output from MQSeries Adapter Builder under Solution Studio. See the MQSeries Adapter Builder documentation for a full definition of adapter.

Contrast with *Business Process Integration Adapter*.

adaptive document. Part of how *Process Choreography* is implemented on Business Flow Manager. The adaptive document includes state machine functionality and persistent data. It can be realized through several approaches.

Different views of the business content are rendered to the end user by the *Interaction Manager*.

application adapter. See *adapter*.

Application Service Provider (ASP). A company that offers subscription services for applications and related services on a pay-per-use basis. ASPs host, manage and maintain applications at their own site and make them available via the Web. This enables smaller companies or those with limited budgets to take full advantage of the latest information technology.

artifacts. The name for the set of all deployable files that make up a solution. Artifacts can be:

- Executables, for example, enterprise beans, Java beans, Java Server Pages, servlets, MQSeries Adapter Offering adapters, and various class files
- Configuration files, for example, MQSeries configuration files that control queues, LDIF (LDAP configuration) files, MQSeries Integrator MRP files, Policy Director map files, and files that control personalization

- Other content, for example, HTML files

You create artifacts by means of Solution Studio and its associated tools such as VisualAge for Java, MQSeries, MQSeries Integrator, MQSeries Adapter Offering, WebSphere Application Server, Partner Agreement Manager, DataInterchange and others. See *solution package*.

ASP. See *Application Service Provider*

Audit Log. The log of messages that are sent between applications and adapters.

Audit Log server. The Business Integrator component that allows the storage and retrieval of audit log information in a DB2 database.

B

B2B. See *business-to-business*.

B2C. See *business-to-customer*.

base machine. The machine that contains the *Topology Repository*, and the Trust and Access Manager facility. Depending on the topology selected, the base machine may also contain other facilities.

BO. See *business object*.

BOD. See *Business Object Document*.

broker. See *message broker*.

business flow. The business processes, at the task level, that drive the business.

Business Flow Manager. The Business Integrator component that controls the flow of business processes. It can be invoked programmatically by Interaction Manager, or via a message from a gateway or an Endpoint. It can invoke Endpoints.

Business Flow Manager provides a platform for:

- *Microflows*
- Data objects
- Workflow enterprise beans that invoke and communicate with MQSeries Workflow
- *JMS Listener*
- *Worker beans*

Business Integrator. The short name for IBM WebSphere Business Integrator.

Business Integrator Log Client. Software that is installed with most of the facilities of Business Integrator, and which allows logging at those facilities.

business object. Data and application objects that persist data for business entities or tasks that are used to populate messages exchanged with other components during run time. Business objects are reusable Enterprise Java Beans (EJB) that typically implement a business entity or task. A business object has both data and function. For example, a request for quotation (RFQ) is a business object that might have a unique identifier, a vendor, and one or more parts.

Business objects are also known as business data objects.

Business Object Document (BOD). A representation of a standard business process that flows within an organization or between organizations. Examples are: add purchase order, show product availability, and add sales order. BODs are defined by the *Open Applications Group* using XML.

business process. The step-by-step flow of information and actions within one organization or between two or more trading partners. A business process can be as simple or complex as needed to meet the business needs of the organization or trading partners. See *task.* There are several types of business processes. See *private process* and *public process*.

Business Process Integration Adapter. Software that enables Partner Agreement Manager to work with the rest of Business Integrator. It packages XML message payloads from a public process into message formats that can be transported by Information Delivery Manager to Endpoints or to the Business Flow Manager. It packages messages from Endpoints or the Business Flow Manager into message formats that can be sent through a public process. The Business Process Integration Adapter manages the correlation information that is necessary to bind public processes in Partner Agreement Manager with business processes in the Business Flow Manager.

business process managers. The Business Integrator components that control Business Integrator. The business process managers are *Trust and Access Manager, Business Flow Manager, Interaction Manager, Information Delivery Manager, and Solution Manager*.

business-to-business (B2B). Electronic commerce where a buyer organization buys goods or services from a merchant organization.

business-to-customer (B2C). Electronic commerce where a consumer buys goods or services from a merchant.

C

capacity unit. A measure of the number of Symmetrical Multiprocessors (SMP) in a machine. This is used to calculate the license requirements when you purchase Business Integrator.

Certificate Authority (CA). A trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The role of the CA is to authenticate the entities (individuals or organizations) involved in electronic transactions. CAs are a critical component in data security and electronic commerce because they guarantee that the two parties exchanging information are really who they claim to be.

channel. In Partner Agreement Manager, an encapsulation of all the processing information needed to send messages to a trading partner's system, and to translate data from a trading partner into Partner Agreement Manager messages. All Partner Agreement Manager installations have the *PAM-to-PAM channel* installed. Available non-Partner Agreement Manager channels include the *RosettaNet* and *XML channel*.

ClearCase repository. In Solution Studio, the storage place for the *solution artifacts*. Rational ClearCase manages version control for all of the artifact files created for each solution.

common systems administration (CSA). Tools and technology used to implement the Product Console Launchpad and solution management framework in Business Integrator. The use of CSA provides a consistent look and feel within the system management consoles of Business Integrator and other IBM products.

correlation identifier. A field in a message that provides a means of identifying related messages. Correlation identifiers are used, for example, to match request messages with their corresponding reply message.

CRM. See *Customer Relationship Management*.

CSA. See *common systems administration*.

Customer Relationship Management (CRM). The systems and infrastructure required to analyze, capture, and share all parts of the customer's relationship with the enterprise. From a strategy perspective, CRM represents a process for measuring and allocating organizational resources to those activities that have the greatest return and impact on profitable customer relationships.

D

DataInterchange. A component of the Business Integrator that performs messaging between the trusted zone and an EDI network or virtual private network. DataInterchange reformats data for transmission via one or more channels, transforms data, supports registration of trading partners, performs auditing and reporting, supports extensive customization, and provides APIs for administrative, logging and command procedures.

DataInterchange can accept messages from the untrusted zone, perform appropriate processing and send the message to Endpoints or to the Business Flow Manager.

Endpoints or the Business Flow Manager can send messages to DataInterchange which cause DataInterchange to execute one or more command procedures to perform one or more actions, for example, to log on to a mailbox, and then to check for messages.

DataInterchange adapter. Software that enables DataInterchange to work with the rest of Business Integrator.

data object. In the Business Integrator programming model, a special type of command that encapsulates access to a data store.

Data Universal Numbering System (DUNS). A system in which internationally recognized nine-digit numbers are assigned and maintained by Dun & Bradstreet to uniquely identify worldwide businesses.

DB2. An IBM relational database management system that is available as a licensed program. Programmers and users of DB2 can create, access, modify, and delete data in relational tables using a variety of interfaces.

DB2 XML Extender. An extension to DB2 that provides data types that let you store XML documents in DB2 databases and functions that assist you in working with these structured documents. Entire XML documents can be stored in DB2 databases as character data or stored as external files but still managed by DB2. Retrieval functions allow you to retrieve either the entire XML document or individual elements or attributes.

DCE. See *distributed computing environment.*

Demilitarized Zone (DMZ). In network security, a network that is isolated from, and serves as a neutral zone between, a trusted network (for example, a private intranet) and an untrusted network (for example, the Internet). One or more secure gateways usually control access to the DMZ from the trusted or the untrusted network.

deployment. The process of making all the elements of a solution available to the run-time system. Compare with *publishing*.

deployment application. Part of the run time that unzips the solution package into the constituent artifacts and that moves the artifacts to the correct location on each machine in the run time environment according to the topology. The deployment application is invoked by the *Solution Deployment Wizard*. See *artifact* and *solution package*.

Digital Certificate. A form of electronic ID. The digital certificate facilitates unique identification of the entity that holds it. It is issued by a Certificate Authority (CA).

distributed computing environment (DCE). A product that assists in networking by providing such functions as authentication, directory service (DS), and remote procedure call (RPC).

DMZ. See *Demilitarized Zone*.

Document Type Definition (DTD). A file associated with XML documents that defines how the markup tags should be interpreted by the application using the document.

DUNS. See *Data Universal Numbering System*.

Dynamic MBean. An MBean that implements the DynamicMBean interface, so called because certain elements of its instrumentation can be controlled at runtime. See *Managed Bean* and contrast with *Standard MBean*.

E

EAI. See *enterprise application integration*.

e-business process integration (e-BPI). A system that enables companies to create, execute, and manage processes that span diverse applications, enterprises, and people, and to manage those processes—as well as the components that support those processes—as a unified, extensive, flexible solution.

Business Integrator is an e-business process integration system.

EDI. See *Electronic Data Interchange*.

EJB. See *Enterprise Java Bean*.

Electronic Data Interchange (EDI). A method of transmitting business information over a network, between trading partners who agree to follow approved national or industry standards in translating and exchanging information.

e-market. Where business-to-business buyers and sellers meet to trade in a virtual market.

Endpoint. A machine within the trusted zone that contains one or more *Endpoint applications*, each with an *application adapter* that allows communication with the Business Integrator system. A Business Integrator system can contain one or more Endpoint machines. Each Endpoint machine contains a single *Endpoint facility*.

Endpoint application. A business application that resides on an Endpoint machine together with an *application adapter*. Endpoint applications are typically called legacy applications or enterprise applications. Examples of such applications are SAP applications.

Endpoint facility. A significant portion of Business Integrator's run time functionality that is installed on each Endpoint machine. An Endpoint facility provides access via one or more adapters to Endpoint applications, and enable messaging within Business Integrator. The Endpoint facility also helps support deployment and management of the solution.

Endpoint machine. Synonymous with *Endpoint*.

element. See *solution element*.

enterprise application integration (EAI). The integration of disparate systems and applications across an enterprise, and the interoperability of complementary systems and applications between enterprises.

Enterprise configuration. The version of Business Integrator used by large enterprises. Enterprise configuration provides capabilities for more complex interactions with trading partners including EDI capability and generalized access to a Web Application server. Compare with *Entry configuration.*

Enterprise Java Bean (EJB). A Java API that defines a component architecture for multi-tier client/server systems. EJB systems allow developers to concentrate on the business architecture of a model, instead of programming the connections between components. EJB systems are platform-independent and object-oriented, and can be implemented into existing systems with minimal recompiling and configuring.

entity bean. A reusable Java component that is built using the Java Beans technology. Entity beans model business concepts that can be expressed as nouns. Entity beans represent data, so a change to an entity bean results in a change on a database. Entity beans are persistent; if the container in which an entity bean is hosted crashes, the entity bean and any remote references survive the crash. Contrast with *session bean*.

Entry configuration. The version of Business Integrator used by small companies and departments of large enterprises. Entry configuration enables trading partners to participate in e-markets with a limited initial investment and relatively low level of complexity. It allows peer-to-peer or spoke-to-hub participation in electronic markets. Compare with *Enterprise configuration*.

EPAC. See *Extended Privilege Attribute Certificate*.

event. In the context of Partner Agreement Manager, a piece of information that comes into Partner Agreement Manager as a message from another source (an enterprise system or business application, for example) and and which triggers a *public process*.

Exception Management server. The Business Integrator component that allows the storage and retrieval of exception information in a DB2 database.

Extended Privilege Attribute Certificate (EPAC). A certificate that contains authorization information specific to the user, for example, details of groups to which the user belongs. EPACs are used to authorize users; that is, to help a server decide whether users should be granted access to resources that the server manages.

Extensible Markup Language (XML). A standard metalanguage for defining markup languages that was derived from and is a subset of Standard Generalized Markup Language (SGML). XML omits the more complex and less-used parts of SGML and makes it much easier to (a) write applications to handle document types, (b) author and manage structured information, and (c) transmit and share structured information across diverse computing systems. The use of XML does not require the robust applications and processing that is necessary for SGML.

extension action. A private process action that communicates, via an adapter, with an external application that is registered with Partner Agreement Manager. You can use an extension action, for example, to get information from an enterprise system or listen for an event in the enterprise system. See also *adapter, private process*.

F

facility. In Business Integrator, an indivisible unit of installation that must be completely installed on one machine. A machine contains one or more facilities. Several facilities installed on a single machine may contain a number of common components, in which case those base components are shared amongst the facilities and not multiply installed.

FDL. See *MQSeries Workflow Definition Language.*

firewall. A functional unit that protects and controls the connection of one network to other networks. The firewall (a) prevents unwanted or unauthorized communication traffic from entering the protected network and (b) allows only selected communication traffic to leave the protected network.

G

gateway. A type of component of the Business Integrator that performs messaging between the trusted zone and the Internet or an EDI network.

There are two gateways:
- Partner Agreement Manager, for the Internet
- DataInterchange, for EDI

Global Secure Toolkit (GSK). A toolkit for managing digital certificates used in implementing Secure Sockets Layer (SSL) security.

H

HTTP. See *Hypertext Transfer Protocol*.

HTTP Server. The component of WebSphere Application Server that provides secure Web Server functionality.

Hypertext Transfer Protocol (HTTP). The protocol used by the Internet to transfer HTML and other information from servers to browsers and other servers.

I

Information Delivery Manager. The Business Integrator component that sends messages:
- Between the gateways and the Business Flow Manager
- Between the Business Flow Manager and Endpoints
- Between the gateways and Endpoints

The messages are typically in a common canonical format, that is, in an application-neutral format such as a *Business Object Document* in XML.

The Information Delivery Manager can perform assured delivery of messages, transformation of data elements and related functionalities, routing of messages, and message brokering. This functionality is performed by MQSeries Adapter Kernel with adapters that you build with

MQSeries Adapter Builder, along with MQSeries for assured delivery, and optionally MQSeries Integrator for message brokering services such as complex routing, data transformation, and data mediation.

instrument. In application or system software, to use monitoring functions to provide performance and other information to a management system.

instrumentation. In application or system software, either (a) monitoring functions that provide performance and other information to a management system or (b) the use of monitoring functions to provide performance and other information to a management system.

Interaction Manager. The Business Integrator component that helps to render a view of a business entity that is appropriate to the role of the end user based on their authorization and the point in the business process. Interaction Manager gets the content of the view that it will render from the Business Flow Manager, as the result of the Business Flow Manager's processing.

How the content is rendered depends on the target device used to view the content. For example, presentation on a Web browser might be different from the presentation on a personal digital assistant.

Interaction Manager is not involved when the content of the view is delivered by one of the gateways.

Internet Inter-ORB Protocol (IIOP). A protocol used for communication between CORBA object request brokers.

J

J2EE Connector Architecture. An architecture for the integration of J2EE products with enterprise information systems. The architecture has two parts: a resource adapter provided by an enterprise information system vendor, and the J2EE product that allows this resource adapter to plug in.

Java 2 Platform, Enterprise Edition (J2EE platform). An environment for developing and deploying enterprise applications. The J2EE platform consists of a set of services, APIs, and protocols that provide functionality for developing multi-tiered, Web-based applications.

Java Authentication and Authorization Service (JAAS). A Java package that enables services to authenticate users and enforce access controls upon them.

Java Cryptography Extension (JCE). A framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. JCE is used by Partner Agreement Manager for certificate-based authentication

Java Database Connectivity (JDBC). An industry standard for database-independent connectivity between the Java platform and a wide range of databases. JDBC provides a call-level API for SQL-based database access.

Java Development Kit (JDK). A software package used to write, compile, debug, and run Java applets and applications.

Java Management Extensions (JMX). A means of doing management of and through Java technology. JMX was developed through the Java Community ProcessSM program, by Sun Microsystems, Inc. and some leading companies in the management field. JMX is a universal, open extension of the Java programming language for management that can be deployed across all industries, wherever management is needed.

Java Message Service (JMS). An API for using enterprise messaging systems such as IBM MQSeries.

Java Runtime Environment (JRE). A subset of the Java Development Kit (JDK) comprising the Java Virtual Machine (JVM), the Java core classes, and supporting files.

Java Secure Socket Extension (JSSE). A Java package that enables secure Internet communications. It implements a Java version of the Secure Sockets Layer (SSL) and Transport Layer Security (TSL) protocols and supports data encryption, server authentication, message integrity, and optionally client authentication. JSSE is used by Partner Agreement Manager for certificate-based authentication

Java Server Pages (JSP). An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to return dynamic content to a client. The template data is typically HTML or XML elements, and the client is often a Web browser.

Java Virtual Machine (JVM). A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

JCE. See *Java Cryptography Extension*

JCX. A deprecated abbreviation for *J2EE Connector Architecture*.

JDBC. See *Java Database Connectivity*.

JDK. See *Java Development Kit*.

JMS. See *Java Message Service*.

JMS Listener. Part of the Business Flow Manager that is started as a WebSphere service. The JMS Listener, in concert with a *worker bean*, decides which enterprise bean in the Business Flow Manager to invoke. The JMS Listener monitors a JMS queue that is associated with a worker bean. It receives the message from the queue and passes it to the worker bean, which then determines the enterprise bean to invoke based on the content of the message.

The JMS Listener is configured through its own XML file, and in WebSphere's Class of Service Naming service.

JMX. See *Java Management Extensions*.

JRE. See *Java Runtime Environment*.

JSP. See *Java Server Pages*.

JSSE. See *Java Secure Socket Extension*.

JVM. See *Java Virtual Machine*.

L

LDAP. See *Lightweight Directory Access Protocol*.

Lightweight Directory Access Protocol (LDAP). An open protocol that (a) uses TCP/IP to provide access to directories that support an X.500 model and (b) does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). Applications that use LDAP (known as directory-enabled applications) can use the directory as a common data store for retrieving information about people or services, such as e-mail addresses, public keys, or service-specific configuration parameters.

Lightweight Third Party Authentication (LTPA). An authentication framework that allows single sign-on across a set of Web servers that fall within an Internet domain.

logical topology view. A view of a topology that shows a tree structure in terms of the facilities of Business Integrator and their base products. See *topology* and *facility*.

M

Managed Bean (MBean). According to the Java Management Extensions (JMX) specification, the Java objects that implement resources and their *instrumentation* are called Managed Beans, or MBeans for short. MBeans must follow the design patterns and interfaces defined in the instrumentation level of the JMX specification. This ensures that all MBeans provide the instrumentation of managed resources in a standardized way. MBeans are manageable by any JMX agent, but they may also be managed by non-compliant agents that support the MBean concept. See also *Java Management Extensions*, *Standard MBean*, and *Dynamic MBean*.

MBean Server. A set of services for handling *MBeans*.

message broker. (1) A set of executing processes hosting one or more message flows. MQSeries Integrator is an example of a message broker. (2) The Business Integrator facility that works with existing messaging transports by adding both routing intelligence and the ability to convert data from one protocol to another. It analyzes a message to determine the application to receive it (rules engine). It then converts the data into the structure that the receiving application requires.

microflow. Part of a *solution*, that is modeled as a Java Service Adapter in MQSeries Adapter Builder. A microflow is deployed as stateless session beans, Java beans, and Java classes, which at run time are part of the Business Flow Manager, running under WebSphere Application Server. A microflow can also can invoke Endpoints, MQSeries Integrator and MQSeries Workflow.

Microsoft Management Console (MMC). An extensible user interface that provides an environment for running management applications structured as components called snap-ins.

middleware. The software that provides the links between applications.

MMC. See *Microsoft Management Console*.

MQSeries. An IBM licensed program that provides reliable message queuing and associated services across a range of platforms.

MQSeries Adapter Builder (MQAB). One of the MQAO set of products, MQAB uses a visual interface to help build an adapter for virtually any application and to build *microflows*.

MQSeries Adapter Kernel (MQAK). One of the MQSeries Adapter Offering set of products that provides common runtime services. In the Business Integrator run time, MQAK, together with MQSeries, and optionally MQSeries Integrator, acts as the Information Delivery Manager.

MQSeries Adapter Offering (MQAO). A set of application integration products that work with MQSeries messaging to reduce the risk, complexity and cost of managing point-to-point application integration. MQAO allows you to create adapters that use a standard interface that remains stable even though the application changes. The interface is typically based on Business Object Documents (BOD), message format standards defined by the Open Applications Group Inc (OAG).

The MQAO is a set of products that includes MQSeries Adapter Builder (MQAB) and MQSeries Adapter Kernel (MQAK).

MQSeries classes for Java Message Service (JMS) (MQ JMS). A set of Java classes, that implement Sun's Java Message Service (JMS) interfaces to enable JMS programs to access MQSeries systems.

MQSeries Integrator (MQSI). A product that works with MQSeries messaging, extending its basic connectivity and transport capabilities to provide a powerful message broker solution driven by business rules.

MQSeries Integrator User Name Server. A component of MQSI that can be used to provide authentication of users and groups performing publish/subscribe operations. At least one of these may be used for each domain, to manage the access paths to resources.

MQSeries Publish/Subscribe. An MQSeries product that allows you to decouple the provider of information from the consumers of the information. An application can send information to a destination managed by MQSeries Publish/Subscribe, which deals with the distribution of the information.

MQSeries Workflow. An MQSeries product that manages *workflow*. MQSeries Workflow is used to design, refine, document, and control business processes.

MQSeries Workflow Definition Language (FDL). The language used to exchange MQSeries Workflow information between MQSeries Workflow systems.

N

node. In Business Integrator one of the points in a *topology view*. Depending on the type of view, a node might correspond to a machine, *facility*, base product, *solution element*, or *solution artifact*.

non-repudiation. In business-to-business communication the ability of the recipient to prove who sent a message based on the contents of the message. This can derive from the use of a digital signature on the message, which links the sender to the message.

O

Object Management Group (OMG). A group of vendors formed for the purpose of creating a standard architecture for distributed objects in networks. The architecture that resulted is the Common Object Request Broker Architecture (Common Object Request Broker Architecture). See also *XML Metadata Interchange*.

Open Applications Group (OAG). A non-profit industry consortium comprised of many prominent stakeholders in the business software component interoperability arena. The OAG defines *Business Object Documents (BOD)*.

P

PAM Proxy Server. Software that resides in the DMZ and which performs access control on inbound messages from business partners to Business Integrator and on outbound messages from Business Integrator to business partners.

PAM-to-PAM channel. A *channel* that is ready-installed in Partner Agreement Manager.

Partner Agreement Connect. A limited licence version of Partner Agreement Manager that allows a customer to participate in a public process defined by a trading partner, but does not allow the authoring of new public processes.

Partner Agreement Manager. The Business Integrator component that implements the *public process* and *trading partner agreements*.

Partner Agreement Manager is a separately purchased, optional component, of both the Entry configuration and Enterprise configuration of Business Integrator.

Partner Agreement View. (1) A products that provides an interface that allows the seamless integration of Partner Agreement Manager with Web applications. (2) The Business Integrator facility that encapsulates Partner Agreement View.

physical topology view. A view of a topology that shows a tree structure in terms of the machines in the topology, the facilities installed on those machines, and their base products. See *topology* and *facility*.

Platform Console. The Business Integrator console used to monitor and manage the business process managers and components of Business Integrator and, through the Solution Deployment Wizard to deploy solution packages onto the runtime system.

point-to-point messaging. Data transmission between two locations without the use of any intermediate display station or computer.

predefined topology. A topology, corresponding to a tested configuration, that can be selected at installation time. A number of predefined topologies are shipped with Business Integrator to allow for different business needs and complexity of solution.

private process. A trading partner's internal sequence of actions for its steps in the *public process*. Although all trading partners in a public process see and agree to its flow, the trading partner that develops a private process is the only one that can ever see it.

process choreography capability. The name of the capability in *Business Integrator* that:

- Aggregates business content that is managed by a variety of underlying processes (also known as *tasks*). One underlying process might be performed by *MQSeries Workflow*, another might be performed by an *Endpoint application*. Processes can be performed by different organizations.

- Provides the output of an underlying process to one or more other underlying processes that might need it as an input.
- Based on the states of the underlying processes, assigns one state to the overall business process.
- Assigns security privileges to each of the participants across the set of processes.
- Dynamically delivers different views of the business content, based on the point in the business process at which the business content is accessed and on the role of the participant who accesses it.
- Manages the life cycle of the business process.

In summary, the process choreography capability coordinates the set of underlying processes that make up a business process, to align the business process with changing business conditions. Many things can happen at any time that can affect what the business process should do next. To align the business process with changing business conditions, the process choreography capability synchronizes information across multiple underlying processes. It maintains the state of the business process apart from the main process.

The process choreography capability is key to understanding how Business Integrator adds value *beyond* the value of the individual underlying products and technologies that are part of Business Integrator.

The *adaptive document* is part of how the process choreography capability is implemented.

process state. The current state of a business process.

Product Console Launchpad. The graphical user interface used to monitor the runtime system of Business Integrator and launch the management consoles of base products of Business Integrator. The Product Console Launchpad provides a wizard to facilitate the addition of new product consoles to the launchpad.

program client. A Business Integrator client that provides for the automated interchange of business documents with trading partners by responding to program requests and maintaining a relationship with requesting programs.

project. In the build time of Business Integrator, the project organizes and contains all *artifacts* of a single solution. The project and its artifacts are stored in the Clear Case server and in the WebSphere Studio Project, on a mapped network drive that can be shared by a team creating a solution together. You employ the project to organize the work-in-progress files during creation of a solution and the artifacts that result when creation is complete. Project is a WebSphere Studio term.

public process. The step-by-step flow of information and actions between two or more trading partners. One trading partner develops the public process, and all trading partners involved review and accept the process before it is implemented. The trading partner that designs a process is its owner. See *private process*.

publishing. The process of preparing a *solution package* using Solution Studio. The solution package is deployed to the runtime system by the *deployment application*. See also *deployment*.

Publishing Wizard. The wizard within Solution Studio that is used to prepare a *solution package*. Contrast with *Solution Deployment Wizard*.

publish/subscribe. See *MQSeries Publish/Subscribe*.

Q

QoS. Quality of service.

queue manager. (1) A program that provides queuing services to applications. It provides an application programming interface to enable programs to access messages on the queues that the queue manager owns. (2) An object that defines the attributes of a particular queue manager.

R

Rational ClearCase. A product used by Solution Studio to enforce version control, provide change management, and as a repository for solution templates, elements, and artifacts. ClearCase is jointly developed between IBM and Rational.

receiver channel. A channel that moves messages from the target to the source machine.

Remote Method Invocation (RMI). A distributed object model for Java program to Java program, in which the methods of remote objects written in the Java programming language can be invoked from other Java virtual machines, possibly on different hosts.

repudiation. Backing out of, or denying taking part in, an e-business transaction.

RMI. See *Remote Method Invocation*.

role-based desktop. A view rendered in a user's browser that provides access to the services the user is authorized to access when they log on to a solution. For example, a user with a defined role of buyer typically has authorization to create and view purchase orders, or a user with a role of administrator typically has authorization to add users and change passwords. In Business Integrator access to the specific services for each user is rendered in the user's browser.

RosettaNet. A non-profit organization that seeks to implement standards for supply chain management transactions on the Internet. The group includes companies such as Microsoft, Netscape, and IBM, and is working to standardize labels for elements such as product descriptions, part numbers, pricing data, and inventory status. The group aims to implement many of its goals through *XML*.

RosettaNet channel. A type of *channel* in Partner Agreement Manager.

S

SCM. See *Supply Chain Management*

SecureWay Directory. A Lightweight Directory Access Protocol (LDAP) cross-platform, highly scalable, robust directory server for security and e-business solutions.

SecureWay Policy Director. A Tivoli product that provides highly available, centralized, authentication, authorization, and user management.

SecureWay Policy Director WebSEAL. The authentication component of SecureWay Policy Director, the product that provides highly available, centralized, authentication, authorization, and user management.

sender channel. A channel that moves messages from the source to the target machine.

service. In the Business Integrator programming model, a set of command operations that is exposed within a public process or a private process.

servlet. An application program, written in the Java programming language, that is executed on a Web server. A reference to a servlet appears in the markup for a Web page, in same way that a reference to a graphics file appears. The Web server executes the servlet and sends the results of the execution (if there are any) to the Web browser.

session bean. An enterprise bean that is created by a client, that usually exists only for the duration of a single client/server session, and which is responsible for managing processes and tasks. A session bean may be transactional, but it is not recoverable if a system crash occurs. Session bean objects can be either stateless or they can maintain conversational state across methods and transactions. Contrast with *entity bean*.

solution. The realization of a business process. The solution is an instance of a solution template. At build time, you build a solution template and employ the project to organize and contain its artifacts. Before deployment the solution is published to create a solution package, and at run time, solutions are deployed, sometimes customized for the run time environment, and then run. When it is running in production, the solution is key to Business Integrators added value.

See *business process, solution template, artifacts. solution package and deployment application*.

solution artifact. See *artifact*.

Solution Console. An extensible application providing a unified view of selected aspects of a solution and solution services at the application level. Solution Console is a web-enabled interface that provides access to data that is persisted in the *Audit Log server*, the *Exception Management server*, and *Trace server* databases.

Solution Deployment Wizard. A wizard called from the Platform Console that invokes the *deployment application* to deploy a *solution package* onto the runtime system. The wizard also allows the redeployment of solution packages that have already been deployed, or partly deployed.

solution elements. The component parts of a solution. An element is a group of artifacts that is installed on one machine. Typical elements of solutions include: Business Processes Model, System Registry, Business Objects, application adapters, Business Flow, Web Clients, and Program Clients.

Solution Manager. The software that provides the infrastructure for monitoring and managing the Business Integrator system. This includes the *Platform Console, Product Console Launchpad* and *Solution Console*, which are used to manage the Business Integrator system and solution.

Solution Manager Client. Software that is installed with most of the facilities of Business Integrator, and which provides the solution management and deployment frameworks. These frameworks consist of *Java Management Extensions (JMX)* and *Managed Beans (MBeans)* to support the management of solutions and deployment of *artifacts*.

solution package. A zip file comprising solution elements, metadata, and scripts created by the publishing process of Solution Studio. The metadata contains information that determines how solution elements will map to machines and facilities in the runtime system. The scripts are used by the *deployment application* in the deployment of the solution to the runtime system.

Solution Services. Services provided by Solution Manager that include audit logging, tracing, events, and exception management.

Solution Studio. The Business Integrator component that is used to define the business processes and assemble the solution.

Solution Studio wizards. Wizards provided by Solution Studio for use in developing solutions. The wizards perform a variety of tasks. For example, many of the tools (products such as MQSeries Adapter Builder) that you use to create the solution *artifacts* are launched by using Solution Studio wizards. These wizards are also used to store the created artifacts in the project folders in the *ClearCase repository*. Solution Studio provides online help for using the wizards.

solution template. In build time, a representation of the business process that the solution is intended to carry out later at run time. In a business domain, such as supply chain management, the solution template defines the following:

- A business process model that defines the business process, for example, in the areas of enterprise integration, planning, forecasting, replenishment, scheduling, order management, and transportation. A typical business process is "purchase order".
- A system registry that defines the run-time repository that contains system configuration and solution-specific information such as users, roles, and trading partner profiles.
- Business objects, which define the distributed objects necessary to support the business process.

- Adapters, which define the connection between gateways, the Business Flow Manager and Endpoints.
- Business flow, which provides a detailed view of the business process.

You model and create the solution template through Solution Studio, and publish the solution template as a solution in a solution package.

Solution templates typically can be reusable assets, and are key to Business Integrator's added value.

solution topology view. A view of a topology that shows a tree structure in terms of the elements and artifacts that make up a solution instance. See *topology*, *facility* and *solution*.

Standard Bean. A class that implements its own MBean interface, See *Managed Bean* and contrast with *Dynamic MBean.*

stateful session bean. A *session bean* that has a conversational state.

state machine. Software that defines one or more states with multiple transitions. Each transition contains a to-state, transition event, conditions, and a set of actions (or commands). For the transition to go to the next state (the to-state), it must successfully execute all the actions defined for the current state. In Business Integrator, a state machine can be realized in a *microflow*.

stateless session bean. A *session bean* that has no conversational state. All instances of a stateless session bean are identical.

Supply Chain Management (SCM). The management of resources, functions, and sequence of processes used by organizations involved in the supply of raw materials and products, and their delivery to manufacturers, wholesalers, retailers, and finally consumers. Business Integrator provides SCM in terms of integrated design, development, and deployment tools for creating solutions that manage the supply of goods and services between supplier and consumer.

T

task. A step in the business process.

topology. A definition of the arrangement of physical machines, together with the software products and components installed on these machines, that make up a Business Integrator runtime environment. A number of *predefined topologies* are shipped with Business Integrator, and one of these is selected at installation time.

Topology Repository. An XML file that stores the details of the machines, facilities and products that make up the topology. The Topology Repository is accessed by Business Integrator runtime components and used to display topology views.

Topology Server. The software that creates, and controls access, to the Topology Repository.

topology type. The identification of an object in the topology repository, for example, computer systems, facilities and products are topology types. Each topology type can have zero, one, or more properties, which can be displayed using the Platform Console or Product Console Launchpad.

topology view. A view of the topology in terms of either the logical, physical, or solution-related elements of the topology. You can use both the Product Console Launchpad and the Platform Console to display a *logical topology view*, *physical topology view*, or a *solution topology view*.

Trace server. The Business Integrator component that allows the storage and retrieval of trace information in a DB2 database.

trading partner agreement (TPA). The formal agreement between trading partners.

transition. A change in state when certain conditions are met.

transmission queue. A queue that stores the messages that are to be sent across a channel.

Trust and Access Manager. The Business Integrator software components that control access to the system and its associated business applications. The Trust and Access Manager provides authorization, authentication, and directory services for Business Integrator. Trust and Access Manager grants users and other business applications access, based on their authorized solutions and roles. The audit and logging capabilities of Trust and Access Manager allow administrators to monitor the system for security breaches.

TPA. See *trading partner agreement.*

U

UML. Unified Modeling Language. A general-purpose notational language for specifying and visualizing complex software, especially object-oriented projects. UML builds on previous notational methods such as Booch, OMT, and OOSE.

UNS. Short for User Name Server. See *MQSeries Integrator User Name Server*.

utilities. In the context of Solution Studio, executable software and associated documentation that can prove useful in building and running solutions. Utility software can be found in the utilities directory in Solution Studio.

W

WebDAV. An abbreviation for Web Distributed Authoring and Version. It is a set of extensions to the HTTP protocol that allows users to collaboratively edit and manage files on remote web servers.

Web Proxy Server. Software that resides in the DMZ and which performs access control on inbound messages from Web clients to Business Integrator

WebSeal. See *SecureWay Policy Director WebSEAL.*

WebSphere. A family of IBM software products that provides a development and deployment environment for basic Web publishing and for transaction-intensive, enterprise-scale e-business applications.

WebSphere Application Server. An e-business application deployment environment built on open standards-based technology. The Advanced Edition is a high-performance EJB server for implementing EJB components that incorporate business logic.

worker bean. An enterprise bean that, in concert with a JMS Listener, decides which enterprise bean in the Business Flow Manager to invoke. See *JMS Listener* for information about how they work together.

Worker beans are configured through the LDAP directory.

worker message bean. Synonymous with *worker bean.*

workflow. The sequence of activities performed in accordance with the business processes of an enterprise. For a full definition, refer to the MQSeries Workflow documentation.

X

X.500. The directory services standard of ITU, ISO, and IEC.

XML. See *Extensible Markup Language*.

XML channel. A type of *channel* in Partner Agreement Manager.

XML Metadata Interchange (XMI). A proposal from the *Object Management Group* that uses the *Extensible Markup Language* (XML) to provide a standard way for programmers and other users to exchange information about metadata (essentially, information about what a set of data consists of and how it is organized). XMI is intended to help programmers using the *Unified Modeling Language* with different languages and development tools to exchange their data models with each other. XMI can also be used to exchange information about data warehouses.

The XMI format standardizes the way in which any set of metadata is described and requires users across many industries and operating environments to see data the same way.

Bibliography

This bibliography lists the books in the IBM WebSphere Business Integrator and associated libraries.

IBM WebSphere Business Integrator library

The Business Integrator library consists of the following books:

- *WebSphere Business Integrator Concepts and Planning, GC34-5960*

  This book introduces the Business Integrator system, providing a high-level system overview, defining the system capabilities, and describing its value to e-businesses. This book also provides the information that you need to plan the installation of Business Integrator.
- *WebSphere Business Integrator Installation Guide, GC34-5961*

  This book is a guide to installing and configuring Business Integrator, It contains information about:
  – Selecting your required topology
  – Installing and configuring the base products and software components of Business Integrator on each machine in the topology
  – Installing and configuring firewalls and proxies
- *WebSphere Studio Business Integrator Extensions Installation Guide, SC34-5962*

  This book is a guide to installing and configuring Solution Studio, It also contains information about setting up clients and servers, and creating projects.
- *WebSphere Business Integrator Run Time*

  This book is a comprehensive guide to the Business Integrator runtime system, providing the following information:
  – Detailed conceptual information about the runtime components of Business Integrator.
  – Deployment of solutions to the runtime system
  – System administration, such as starting and stopping software components and base products, defining users, and using the Exception Console.
  – General problem determination information, including how to trace and debug, and information on obtaining help from technical support

- *WebSphere Business Integrator Messages*

  This book lists the error messages that are produced by Business Integrator and provides references to the documentation for the messages of base products.
- *WebSphere Studio Business Integrator Extensions Developer's Guide*

  This book describes how to create a Business Integrator solution, beginning with the solution design phase, to the solution implementation phase, and finally the solution deployment phase using a sample business problem. This book also provides procedures for assembling a Business Integrator solution in the run-time environment and a description of how to use the Solution Studio for solution design and implementation.
- *WebSphere Business Integrator DataInterchange for Windows NT User's Guide, SC34-5963*

  This book is a guide to installing and using DataInterchange, in the Business Integrator environment.

You can find the latest versions of the books at the following Web site:

`http://www-4.ibm.com/software/webservers/btobintegrator/`

This site contains links to the Web sites of the underlying products of IBM WebSphere Business Integrator.

Related documentation

The `utilities` subdirectory on the Documentation CD contains documentation about utilities that can prove useful in building and running solutions. This documentation is not available on the IBM WebSphere Business Integrator Web site.

WebSphere Business Integrator also provides a number of external application programming interfaces (API). HTML documentation that is generated using the Javadoc tool is provided for these APIs. For a list of the APIs, refer to the *WebSphere Business Integrator Run Time* book.

WebSphere Partner Agreement Manager library

The Partner Agreement Manager Version 2 Release 1 library consists of:

- *Partner Agreement Manager Installation Guide*, GC34-5964
- *Partner Agreement Manager Administrator's Guide*
- *Partner Agreement Manager User's Guide*
- *Partner Agreement Manager Adapter Developer's Guide*
- *Partner Agreement Manager Script Developer's Guide*
- *Partner Agreement Manager API Guide*
- *Partner Agreement Manager Adapters for MQSeries User's Guide*
- *Partner Agreement View User's Guide*, GC34-5965
- *WebSphere Partner Agreement Manager Business Process Integration Adapter Guide*.

DataInterchange library

The DataInterchange Version 3 Release 1 library consists of:

- *DataInterchange Client User's Guide*, SB34-2010
- *DataInterchange Administrator's Guide*, SB34-2002
- *DataInterchange Installation Guide*, GB09-8070
- *DataInterchange Messages and Codes*, SB34-2000
- *DataInterchange Programmer's Reference*, SB34-2001

Other Libraries

You can find important information in the libraries of the following products:

- DB2® UDB
  - *IBM DB2 Universal Database Quick Beginnings Version 6.1* , S10J-8149
- MQSeries®
  - *MQSeries for Windows NT Quick Beginnings*, GC34-5389
  - *MQSeries System Administration*, SC33-1873
  - *MQSeries Using Java*, SC34-5456
  - *MQSeries MQSC Command Reference*, SC33-1369
  - *MQSeries Queue Manager Clusters*, SC34-5349
  - *MQSeries Integrator Introduction and Planning*, GC24-5599
  - *MQSeries Workflow Getting Started with Buildtime*, SH12-6286
  - *MQSeries Workflow Getting Started with Runtime*, SH12-6287
  - *MQSeries Adapter Kernel for Multiplatforms: Quick Beginnings*, GC34-5855
  - *MQSeries Adapter Kernel for Multiplatforms: Problem Determination Guide*, GC34-5897

- *MQSeries Adapter Builder for Windows NT: Using the Control Center*, GC34-5882
- SecureWay®
  - *SecureWay Policy Director Up and Running*, SCT6-3KNA
  - *SecureWay Policy Director Base Administration Guide*
  - *SecureWay Firewall User's Guide*, CG31-8658
- VisualAge®
  - *VisualAge Java, Enterprise Edition Getting Started*
  - *VisualAge C++ Professional for Windows NT Getting Started*
- WebSphere™ Application Server
  - *Introduction to WebSphere Application Server*, SC09-4430

Index

Part Number: BPIAAC00
Program Number: 5724-A78

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

(1P) P/N: BPIAAC00