IBM Integration Bus 9.0.0
Version 1 Release 0

IBM Integration Bus 9.0.0 Overview

Note
Before using this information and the product it supports, read the information in "Notices" on page 121.

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

Contents

Figures v

Chapter 1. IBM Integration Bus overview 1

Chapter 2. IBM Integration Bus introduction 3

Chapter 3. What's new in Version 9.0?. . 5
What else is new if you are migrating from WebSphere Message Broker Version 7.0 ? 10
What else is new if you are migrating from WebSphere Message Broker Version 6.1?. 20
What's new in IBM Integration Bus for WebSphere Enterprise Service Bus users? 39

Chapter 4. Name changes in IBM Integration Bus Version 9.0 49

Chapter 5. IBM Integration Bus technical overview. 51
Create the broker environment 52
Develop applications 53
Deploy applications to the runtime environment . . 53
Publish/Subscribe 53
Further information 54
IBM Integration features 54
IBM Integration Toolkit 58
The IBM Integration Toolkit 58
Accessing context-sensitive help 58
IBM Integration Toolkit perspectives 59
Editors 59
Resources 60
Development repository 65
The IBM Integration Bus environment 65
Operation modes 66
System management interfaces 71
Execution groups 71
The IBM Integration API 72
IBM Integration Explorer 73
The IBM Integration Explorer 73
Accessing context-sensitive help 74
IBM Integration Bus web user interface 75
External systems and resources 76

Chapter 6. Business scenarios 79
Healthcare integration scenarios 79
Mobile integration scenarios 80
Featured scenario: .NET service enablement for mobile applications 80
More scenarios 82
.NET integration scenarios 82
Featured scenario: .NET service enablement for mobile applications 82
More scenarios 83
Mergers and acquisitions scenario 84

Chapter 7. Samples 87
Creating the Default Configuration 96
What the Default Configuration wizard creates 96

Chapter 8. Glossary of terms and abbreviations 99

Chapter 9. Accessibility features for IBM Integration Bus. 119

Notices 121
Programming interface information 123
Trademarks 123

Sending your comments to IBM . . . 125

Figures

Chapter 1. IBM Integration Bus overview

This section provides introductory information to help you get started with IBM® Integration Bus:

- Chapter 2, “IBM Integration Bus introduction,” on page 3
- Chapter 3, “What's new in Version 9.0?,” on page 5
- Technical overview
- Scenarios
- Start here
- Chapter 7, “Samples,” on page 87
- Legal information
- Chapter 8, “Glossary of terms and abbreviations,” on page 99
- Chapter 9, “Accessibility features for IBM Integration Bus,” on page 119

Chapter 2. IBM Integration Bus introduction

You can use IBM Integration Bus to connect applications together, regardless of the message formats or protocols that they support.

This connectivity means that your diverse applications can interact and exchange data with other applications in a flexible, dynamic, and extensible infrastructure. IBM Integration Bus routes, transforms, and enriches messages from one location to any other location:

- The product supports a wide range of protocols: WebSphere® MQ, JMS 1.1, HTTP and HTTPS, Web Services (SOAP and REST), File, Enterprise Information Systems (including SAP and Siebel), and TCP/IP.
- It supports a broad range of data formats: binary formats (C and COBOL), XML, and industry standards (including SWIFT, EDI, and HIPAA). You can also define your own data formats.
- It supports many operations, including routing, transforming, filtering, enriching, monitoring, distribution, collection, correlation, and detection.

Your interactions with IBM Integration Bus can be considered in two broad categories:

- Application development, test, and deployment. You can use one or more of the supplied options to program your applications:
  - Patterns provide reusable solutions that encapsulates a tested approach to solving a common architecture, design, or deployment task in a particular context. You can use them unchanged or modify them to suit your own requirements.
  - Message flows describe your application connectivity logic, which defines the exact path that your data takes in the broker, and therefore the processing that is applied to it by the message nodes in that flow.
  - Message nodes encapsulate required integration logic, which operates on your data when it is processed through your broker.
  - Message trees describe data in an efficient, format independent way. You can examine and modify the contents of message trees in many of the nodes that are provided, and you can supply additional nodes to your own design.
  - You can implement transformations by using graphical mapping, Java™, PHP, ESQL, and XSL, and can make your choice based on the skills of your workforce without having to provide retraining.
- Operational management and performance.
  - An extensive range of administration and systems management options are available for developed solutions.
  - A wide range of operating system and hardware platforms are supported.
  - A scalable, highly performing architecture, based on requirements from traditional transaction processing environments.
  - Tightly integrated with software products, from IBM and other vendors, that provide related management and connectivity services.

  IBM Integration Bus is available in several modes, so that you can purchase a solution that meets your requirements.

Application development

Your message processing applications, which you can run on more than 30 industry platforms, can connect to the broker by using one of the supported protocols already listed. Platforms from IBM, Microsoft, Oracle, and others are supported.

Diverse applications can exchange information in widely differing formats, with brokers handling the processing required for the information to arrive in the right place in the correct format, according to the rules that you have defined. The applications need only to understand their own formats and protocols, and not standards used by the applications to which they are connected.

Applications also have much greater flexibility in selecting which messages they want to receive, because you can apply filters to control the messages that are made available to them.

IBM Integration Bus provides a framework that contains a wide variety of supplied, basic, functions along with user-defined enhancements, to enable rapid construction and modification of message processing rules.

Your applications can be integrated by providing message and data transformations in a single place, the broker. This integration helps to reduce the cost of application upgrades and modifications. You can extend your systems to reach your suppliers and customers, by meeting their interface requirements within your brokers. This ability can help you to improve the quality of your interactions, and allow you to respond more quickly to changing or additional requirements.

Messages are manipulated according to the rules that you define by using the IBM Integration Toolkit.

Operational management

IBM Integration Bus supports a choice of interfaces for operation and administration of your brokers:

- The IBM Integration Toolkit
- The IBM Integration Explorer is a graphical user interface, based on the WebSphere MQ Explorer, for administering your brokers
- Applications that use the IBM Integration API (also known as the Integration API)
- A comprehensive set of commands, that you can run interactively or by using scripts
- The Representational State Transfer API (REST) allows development of administrative applications without the need to install client software and web browsers can administer brokers through a user interface.

IBM Integration Bus builds on the WebSphere MQ product, which provides assured, once-only delivery of messages between the applications. WebSphere MQ is included when you purchase IBM Integration Bus.

IBM Integration Bus is complemented by a wide variety of other IBM products such as Tivoli® Composite Application Manager for SOA, WebSphere Service Registry and Repository (WSRR), WebSphere Process Server, and WebSphere Transformation Extender (WTX).

Chapter 3. What's new in Version 9.0?

Learn about the main new functions in IBM Integration Bus Version 9.0.

IBM Integration Bus is a compatible evolution of WebSphere Message Broker that is designed to incorporate features that are found in WebSphere Enterprise Service Bus. IBM Integration Bus provides a universal integration capability that addresses a wide range of integration scenarios. These scenarios include web services such as SOAP and REST, messaging, database, file, ERP systems, mobile, physical devices, email, custom systems and more. To reflect these new capabilities, the product name and some of the resource names have changed. For more information, see Chapter 4, “Name changes in IBM Integration Bus Version 9.0,” on page 49.

For details of the new features in IBM Integration Bus Version 9.0, see the following sections.

- “Simplicity and productivity”
- Support for .NET
- Graphical mapping
- “Web services” on page 7
- “Universal connectivity for SOA” on page 8
- Dynamic operation management
- “High performance and scalability” on page 9
- “Platforms and environments” on page 9
- “Troubleshooting and support” on page 10

If you are migrating from WebSphere Message Broker Version 7.0. also see “What else is new if you are migrating from WebSphere Message Broker Version 7.0 ?” on page 10.

If you are migrating from WebSphere Message Broker Version 6.1, also see “What else is new if you are migrating from WebSphere Message Broker Version 6.1?” on page 20.

If you are migrating from WebSphere Enterprise Service Bus, also see “What's new in IBM Integration Bus for WebSphere Enterprise Service Bus users?” on page 39.

Simplicity and productivity

Business rules

You can use IBM Integration Bus to write business rules by using natural language, so that they can be read easily by business users (for example, a business analyst). In IBM Integration Bus, you create a *decision service*, which is a collection of rules that are used to process a message. A DecisionService node executes those business rules to provide operations like routing, validation, and transformation. You can either write rules in the IBM Integration Toolkit, or import rules from IBM Operational Decision Manager. You can also retrieve rules from an external IBM Operational Decision Manager repository.

The DecisionService node allows IBM Integration Bus to call business rules that run on a component of IBM Decision Server that is provided with IBM Integration Bus. The IBM Integration Bus license entitles you to use this

component only through the DecisionService node and only for development and functional test. To use the IBM Decision Server component beyond development and functional test, you must purchase a separate license entitlement for either IBM Decision Server or IBM Decision Server Rules Edition for Integration Bus.

For more information, see Business rules.

Discovery of services
Use IBM Integration Toolkit to discover services that you can use to connect to a database or to WebSphere MQ.

Database Service
Use the Database Service to make database operations accessible both within a message flow, and to external applications calling a message flow.

MQ Service
Use the MQ Service to connect to a WebSphere MQ application by discovering resources from existing WebSphere MQ Queue Managers.

For more information, see Database Service and MQ Service.

Message flow statistics in the web user interface
You can use the IBM Integration web user interface to start and stop the collection of message flow statistics, and to display snapshot statistical data in graphical and tabular formats. For more information, see Analyzing message flow performance.

Statistics multiple output formats
You can specify one or more of the following output formats for message flow accounting and statistics data:
- XML
- JSON
- SMF
- User trace

For more information, see Output formats for message flow accounting and statistics data.

Support for .NET

New patterns to rapidly integrate Microsoft applications
IBM Integration Bus Version 9.0 delivers additional productivity aids, enabling Microsoft applications to be rapidly integrated as part of a heterogeneous integration solution. Prebuilt patterns for Microsoft Dynamics CRM allow transformation and synchronization of client data with another CRM application. This facilitates sharing of the state of business relationships and transactions consistently across an organization, while preserving data quality.

- A prebuilt pattern assists with the integration of Microsoft Dynamics CRM with a SAP system. Updates to customer data from create, read, update, and delete operations using a SAP BAPI can be easily reflected in a matching Dynamics CRM Account Entity. See Microsoft Dynamics CRM Account Entity output: Static BAPI input.
- A prebuilt pattern builds upon the first by allowing a Microsoft Dynamics CRM application to be updated by using various other

mechanisms, including WebSphere MQ messages, HTTP, or flat files. See Microsoft Dynamics CRM Account Entity output: Dynamic transport input.

.NETInput node
The .NETInput node provides an easy way to drive message flows that can receive data from other applications that have Microsoft .NET or Component Object Model (COM) interfaces. This feature builds upon the Microsoft .NET Framework support that is introduced in WebSphere Message Broker Version 8.0, that you can use to host and run .NET applications and code inside a broker. The .NETInput node provides C#, Visual Basic, and Visual F# templates for Microsoft Visual Studio, which when implemented, connect and retrieve the data for the node. For example, the .NETInput node can be used to retrieve messages from MSMQ.

For information about configuring and using the .NETInput node, see .NETInput node and Using .NET.

You can also create a template from an instance of a .NETInput node in a message flow. The result is a Cloned node that has its own icon, properties and runtime implementation, which can be easily exported and shared with other IBM Integration Toolkit developers. For more information, see Cloning a .NETInput node.

.NETInput node sample
A sample is provided to demonstrate how to use the .NETInput node. For more information, see .NETInput node to MSMQ. You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.

Graphical mapping

Database Routine transform

The Database Routine transform is provided to enable calling a stored procedure from a graphical data map. You can then create mappings from the output parameters, return values, and result sets from your Database Routine by using the Return transform.

For more information, see Database Routine and Calling a stored procedure from a map.

Web services

Web services enhancements improve security and reliability.

Internationalized Domain Names (IDNs)
Non-English host names can be used in web browsers to accommodate non-native English speakers as a significant group of Internet users.

SSL client authentication selection key
SSL-based nodes can specify a key alias for use by the JSSE layer for a key in the keystore to be used for a specified connection.

Inbound and Outbound support for multiple personal certificates
As a sender, you can dynamically specify which certificate is used to identify IBM Integration Bus to an external application. As a listener, you can dynamically select the certificate to apply to an incoming request.

HTTP BasicAuth support
HTTP BasicAuth support is enhanced to more closely adhere to the BasicAuth specification.

Support for multiple Kerberos Service Principals in a single broker
The previous restriction of one Kerberos Service Principal per broker host is removed. You can now define a Kerberos Service Principal per broker and per execution group.

Support for Certificate Revocation Lists (CRLs) on SSL requests
You can configure IBM Integration Bus to check CRLs when it is acting as an HTTP provider. For more information, see Working with certificate revocation lists

Universal connectivity for SOA

Integration with IBM Business Process Manager Standard Edition
You can use IBM Business Process Manager with IBM Integration Bus. You can make use of patterns and samples from IBM Integration Bus to implement IBM Business Process Manager (BPM) Integration services.

For more information, see Working with IBM Business Process Manager.

IBM Integration administration for WebSphere Application Server
An administrator for WebSphere Application Server can start or stop integration servers (execution groups), applications, integration services, and message flows by using the WebSphere Integrated Solutions Console. The role-based security model that IBM Integration provides determines whether the actions are authorized.

Coordinated transactions for CICS® requests
CICSRequest nodes support one-phase commit coordinated transactions, which means that multiple requests to a CICS server can be handled as part of the same transaction and will be committed or rolled back together depending on the outcome of the message flow. For more information, see CICSRequest node and Message flow transactions.

CICS activity log
The CICS activity log provides a high-level overview of the recent interactions between IBM Integration Bus and CICS. For more information, see CICS Activity log.

Application-to-application integration
IBM Integration Bus Version 9.0 can receive events that are emitted from WebSphere Application Server when Web Service client requests are intercepted by a Local Mapping Service. See Record and replay.

IBM MessageSight integration
IBM MessageSight provides a high-volume, high-capacity messaging capability for mobile and enterprise applications and is designed for use on the edge of an enterprise network. IBM Integration Bus Version 9.0 can connect to IBM MessageSight by using the existing JMS nodes. You can then use IBM Integration Bus Version 9.0 to route data between large numbers of external devices and enterprise applications. IBM Integration Bus Version 9.0 includes a pattern that demonstrates how you can use this integration capability. For information about the pattern, see IBM MessageSight: Inbound event filter

Dynamic operation management

Workload management

Allows system administrators to monitor and adjust the speed that messages are processed, as well as controlling the actions taken on unresponsive flows and threads.

For more information, see Workload management.

Execution group user IDs on z/OS®

On z/OS, you can specify an alternative user ID to run an execution group so that it accesses resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID. For more information, see Execution group user IDs on z/OS .

High performance and scalability

Host WebSphere eXtreme Scale container servers in multi-instance brokers

When you require high availability, you can host multiple container servers in a multi-instance broker. If the active instance of the broker fails, the global cache switches to use the container servers in the standby instance of the broker. A new XML policy file is provided that demonstrates this configuration. You can now specify multiple listener hosts for a multi-instance broker.

For more information, see Configuring the global cache for multi-instance brokers.

Scale mode in Standard Edition

If you purchase the Standard Edition license, you are entitled to run brokers in either Standard mode or Scale mode. For more information about these modes, see “Operation modes” on page 66.

Platforms and environments

Additional features that detail improved control over message processing, updated ODBC database driver support, and support for Standard Edition on z/OS.

Conversion of WebSphere Enterprise Service Bus resources

You can develop IBM Integration Bus applications and integration services by using WebSphere Enterprise Service Bus development resources. You can use the WebSphere ESB conversion tool to help you convert WebSphere Enterprise Service Bus development resources.

For more information, see Developing integration solutions by using WebSphere Enterprise Service Bus resources.

The WebSphere ESB conversion tool accelerates the conversion of WebSphere Enterprise Service Bus resources to IBM Integration Bus resources while reducing management overhead and extending the reusability of service components.

For more information, see The WebSphere ESB conversion tool.

Aggregation sub-second timeouts

Aggregated message processing now allows the setting of the timeout property to one decimal place, resulting in timeout intervals of less than one second if required.

For more information, see Setting timeout values for aggregation.

Developer Edition is restricted to processing one message per second
Developer Edition is limited to one message (transaction) per second at the message flow level.

For more information, see “Operation modes” on page 66.

DataDirect V7.0 driver support
On Windows, Linux and UNIX systems, IBM Integration Bus Version 9.0 ships with the DataDirect V7.0 ODBC drivers.

For more information, see Enabling ODBC connections to the databases.

Standard Edition now supported on z/OS
IBM Integration Bus for z/OS is now available in Standard Edition.

For more information, see License requirements.

Troubleshooting and support

Verify the ODBC environment on Linux and UNIX systems
The **mqsicvp** command is run automatically when you start a broker by using the **mqsistart** command. The command checks that the broker environment is set up correctly. This checking has been enhanced to verify that the ODBC environment is configured correctly on Linux and UNIX systems. When you run this command from the command line on Linux and UNIX systems, it also validates the connection to all data sources that are listed in the `odbc.ini` file that have been associated with the broker by using the **mqsisetdbparms** command.

For more information, see **mqsicvp** command.

What else is new if you are migrating from WebSphere Message Broker Version 7.0 ?

Learn about additional functions in IBM IBM Integration Bus Version 9.0 that are new if you are migrating from WebSphere Message Broker Version 7.0 or Version 6.1.

The following features and capabilities were introduced in WebSphere Message Broker Version 8.0 and are included in IBM Integration Bus Version 9.0. For information about the features and capabilities that are introduced in IBM Integration Bus Version 9.0, see Chapter 3, “What's new in Version 9.0?,” on page 5.

- “Simplicity and productivity”
- “High performance and scalability” on page 12
- “Application development enhancements” on page 13
- “Deployment and administration enhancements” on page 15
- “Web services enhancements” on page 16
- “Universal connectivity for SOA” on page 16
- “Dynamic operation management” on page 17
- “Troubleshooting and support” on page 18
- “Platforms and environments” on page 19

Simplicity and productivity

The development, deployment, management, and migration of IBM Integration Bus solutions is simplified.

Web administration

You can use the IBM Integration Bus web user interface to administer broker resources. The web user interface enables web users to access broker resources through an HTTP client, and provides broker administrators with an alternative to the IBM Integration Explorer for administering broker resources. The web user interface listener is enabled by default for new brokers.

For more information, see Administering brokers using the web user interface.

RESTful API support

IBM Integration Bus supports the REST management API for broker administration.

For more information, see Representational State Transfer (REST) API.

Converting projects to applications and libraries

If you imported resources from a previous version of WebSphere Message Broker, you can convert all or some of your projects automatically to applications and libraries. A refactoring wizard applies rules to decide how to convert your resources. The wizard identifies errors that would prevent successful conversion, and provides fixes for those errors, where possible.

For more information, see Converting a project to an application or library.

Service creation

Web services are now a first-class artifact in IBM Integration Bus. You can create web services in the new Services editor, and implement the operations as message flows.

For more information, see Developing integration solutions by using integration services.

Export mapping information

You can now export mapping information from Compute nodes, for use in impact analysis and data lineage operations in products such as IBM InfoSphere® Metadata Workbench. See Exporting mapping information from Compute nodes.

New `jdbcProviderXASupport` property

An optional property that controls whether the broker connects to a database server by using XA Protocol. For more information, see Setting up a JDBC provider for type 4 connections.

Creating multi-instance brokers with no domain controller restrictions

You can now create a multi-instance broker without installing Windows on a domain controller. You can also create a multi-instance broker and a multi-instance message queue at the same time.

For more information, see Creating a multi-instance broker and Creating a multi-instance broker and a multi-instance queue manager .

Analyze and filter information in XML files

The Data Analysis perspective analyzes and filters information in XML files. You can use this analysis to create Data Analysis tools to transform your data.

For more information, see Data Analysis.

Handling large messages in Java

When you design a message flow that handles large messages that are made up of repeating structures, you can code Java methods that help to reduce the storage load on the broker.

For more information, see Working with large input messages to propagate multiple output messages.

High performance and scalability

The use of a wide range of hardware, software, and virtualized environments is facilitated.

WebSphere eXtreme Scale global cache

A global cache is a repository for data that you want to reuse. The cache facilitates sharing of data across processes, and eliminates the need for an alternative solution, such as a database. The global cache is embedded in the broker. The cache has a default topology and can be used immediately without any configuration. However, you can turn off the default configuration and set properties explicitly for each execution group.

For more information, see Data caching overview.

Developer Edition

IBM Integration Bus Developer Edition is a full function version of the product, which you can use for evaluative purposes. You can download the Developer Edition at no charge and you are free to use it for as long as you require, within the terms of the license.

Developer mode is introduced with the Developer Edition. All brokers that you create in the Developer Edition are created in Developer mode by default.

There is no longer a Trial Edition for IBM Integration Bus.

Developer Edition is restricted to processing one message per second.

For more information, see “Operation modes” on page 66.

External WebSphere eXtreme Scale grids

An embedded global cache was provided in the previous version as a repository for data that you want to reuse. In addition to the grid that is available (as the embedded global cache) in IBM Integration Bus, you can now integrate with WebSphere eXtreme Scale grids that are running elsewhere. You can work with multiple external grids, and the embedded grid, at the same time. You can also enable SSL for connections to external WebSphere eXtreme Scale grids.

Enable the new capability by using the **-f** parameter on the **mqsichangebroker** command, as described in **mqsichangebroker** command. You must stop the cache before you can enable the capability for the brokers.

For more information, see WebSphere eXtreme Scale grids.

Using a domain name to identify grids

WebSphere eXtreme Scale clients use a domain name to identify and distinguish between embedded grids. Only WebSphere eXtreme Scale servers with the same domain name can participate in the same grid. If you do not specify a domain name, the broker creates a name that is based on the server names of the catalog servers.

By default, each server starts with a domain name that is derived by the broker. In previous versions of WebSphere Message Broker, the domain name for all servers in all embedded caches was an empty string. Servers in different domains cannot collaborate in the same grid. Therefore, for a cache that spans more than one broker, you must enable the new capability for these brokers at the same time.

Brokers on IBM Integration Bus Version 9.0 can still collaborate in the same grid as brokers on WebSphere Message Broker Version 8.0.0.2 or earlier. In this scenario, the capability should not be enabled on the WebSphere eXtreme Scale servers, and then the servers continue to use an empty string as the domain name; servers report a BIP7140 warning message to the system log with this information.

Enable the capability by using the **-f** parameter on the **mqsichangebroker** command, as described in **mqsichangebroker** command.

For more information, see Configuring the embedded global cache by using commands.

Removing data from the global cache

You can specify how long data exists in the global cache by setting a *time to live* value. You specify this value when you get an MbGlobalMap object. The value applies to all cache entries that are created by using that MbGlobalMap object in that instance of the JavaCompute node. After the specified time, the affected data is removed from the global cache automatically.

Enable the new capability by using the **-f** parameter on the **mqsichangebroker** command, as described in **mqsichangebroker** command. You must stop the cache before you can enable the capability for the brokers.

For more information, see Embedded global cache.

Scale mode

Scale mode provides support for unlimited execution groups and a defined subset of nodes. In Scale mode, you can create multiple brokers and deploy message flows containing the supported nodes to your execution groups. WebSphere Enterprise Service Bus customers can obtain a transfer license, which enables them to migrate to IBM Integration Bus and to create and run brokers in Scale mode.

For more information, see “Operation modes” on page 66.

New `timeoutThreads` property

An optional property that assigns additional processing threads to enable processing of timed out aggregation messages in the AggregateReply node. For more information, see Processing timed out aggregation messages.

Application development enhancements

Application development is simplified.

Applications and libraries

Applications and libraries introduce a new way of creating and managing resources when you are developing and deploying applications, and during operational management.

For more information, see Applications and libraries.

IBM Integration API for developing message flow applications

Use the IBM Integration API to write Java programs that create or modify message flows. You do not need to install the IBM Integration Toolkit to run programs that use the IBM Integration API.

For more information, see Developing message flow applications by using the IBM Integration Java API.

User-defined pattern enhancements

Create projects in your user-defined patterns in which pattern users can create customizations that are not overwritten when a pattern instance is regenerated. For further information, see Configuring a project used in a user-defined pattern.

Create table parameter types for your user-defined patterns. For more information, see Using tables for pattern parameters.

Improved graphical mapping

The improved graphical mapping is high-performing, scalable, and dynamic.

Graphical data maps enable you to create transformations that take an input message assembly and build a required output message assembly. For more information, see Using graphical data maps.

DFDL support

- **New DFDL message modeling support**

  You can use the Data Format Description Language (DFDL) support in IBM Integration Bus to model the structure of general text and binary formatted messages in a way that is independent of the message format.

  For more information, see Data Format Description Language (DFDL).

- **New DFDL schema editor**

  You can use the DFDL schema editor to create, edit, and test DFDL message models within the IBM Integration Toolkit.

  For more information, see DFDL schema editor.

- **New DFDL domain**

  You can use the DFDL domain to parse and write a wide variety of message formats that have a DFDL message model. The DFDL domain is primarily intended for non-XML message formats.

  For more information, see DFDL parser and domain.

.NET support

- **New .NETCompute node**

  You can use the Microsoft .NET Framework (.NET) support in IBM Integration Bus to host and run .NET applications and code from inside an Execution Group.

  The .NETCompute node routes or transforms messages by using any Common Language Runtime (CLR) compliant .NET programming language, such as C#, Visual Basic (VB), F# and C++/CLI (Common Language Infrastructure). By using this node, the broker can interact with other applications that have .NET or Component Object Model (COM) interfaces and perform tasks such as message enrichment, by obtaining data from these applications.

  For information about configuring and using the .NETCompute node, see .NETCompute node and Using .NET.

- **ESQL enhancements**

  .NET methods can be called directly from ESQL.

  For more information, see CREATE FUNCTION statement and CREATE PROCEDURE statement.

Deployment and administration enhancements

Additional features provide better information and control of operations.

Resource statistics for JMS

View the JMS statistics to view the number of JMS connections that are used and messages that are processed by nodes that use JMS transport. For more information, see Resource statistics.

Activity Logs

Use the new Activity logs to get an overview of recent activities in your message flows and associated external resources.

For more information, see Using Activity logs.

Deploying flows and applications in stopped state

You can specify how a message flow or application is started after it is deployed, or after the broker, execution group, or containing application is restarted. You can choose to start an application or flow manually, or for it to be started automatically. You can also choose to maintain the existing state of an application or library.

For more information, see Configuring the start mode of flows and applications at development time.

Deployable subflows

Create subflows that you can deploy to a broker as individual resources. Use the deployable subflow in more than one message flow application so that when you change the subflow and redeploy it, all your message flow applications use the updated subflow.

For more information, see Subflows.

Deployable ESQL

Create ESQL files that you can deploy to a broker as individual resources. Use the code from the deployed ESQL file in more than one message flow application so that when you change the ESQL file and redeploy it, all your message flow applications that reference the ESQL code are updated.

For more information, see Deploying an ESQL file.

Dynamic configurable services

Your changes to most types of configurable service will take effect from the next time a message flow that uses the configurable service is called. For some types of configurable service, for which it is explicitly stated in the documentation, you must stop and start the execution group for the change to take effect.

For more information, see Configurable services.

Record and replay

For audit purposes or problem determination, you might want to keep a record of messages that pass through a message flow. You can record those messages in a database, then view them. For more information, see Recording, viewing, and replaying data.

Web services enhancements

Web services enhancements improve security and reliability.

Web Services Reliable Messaging
Administrators can configure message flows and nodes to use WS-RM (Web Services Reliable Messaging) for inbound and outbound SOAP messages. For more information, see Web Services Reliable Messaging.

Internationalized Domain Names (IDNs)
Non-English host names can be used in web browsers to accommodate non-native English speakers as a significant group of Internet users.

SSL client authentication selection key
SSL-based nodes can specify a key alias for use by the JSSE layer for a key in the keystore to be used for a specified connection.

Universal connectivity for SOA

Additional nodes, configurable services, and other capabilities expand the interaction of the broker with other products.

JMSReceive node
Use the new JMSReceive node to consume or browse messages from a JMS queue in the middle of a message flow.

For more information, see JMSReceive node.

Improved integration with WebSphere Application Server

You can view your broker resources, including web services, by using IBM Integration Bus administration for WebSphere Application Server. See Administering integration nodes from WebSphere Application Server.

DFDL improvements

You can import C header files; see Creating a DFDL schema file by using the New Message Model wizard. Field length prefixes are supported. A new industry sample is provided, consisting of a DFDL schema for ISO8583, and a message flow for transforming ISO8583 messages to XML, and vice versa.

Graphical Data Mapping Enhancements

You can modify database content from your Graphical Data Maps; see Modifying data in a database by using mapping .

You can also convert a message map from a previous version of IBM Integration Bus to a graphical data map; see Converting a message map from a `.msgmap` file to a `.map` file.

SOAP nodes can use the broker-wide HTTP listener

You can configure your execution groups so that the SOAP nodes use the broker-wide HTTP listener to process HTTP messages, rather than the execution group listener. Using the broker-wide listener can simplify the administration of your broker configuration. For more information, see HTTP listeners.

HTTP asynchronous request-response

Use new HTTP asynchronous nodes to call an HTTP web service and receive an asynchronous response without blocked waiting. For more information, see Using HTTP asynchronous request-response.

You can also use HTTP asynchronous response-request behavior instead of WS-Addressing with the SOAPAsyncRequest node to make HTTP requests and receive an asynchronous response. For more information, see Choosing asynchronous behavior for the SOAPAsyncRequest node.

Enhanced support for external web servers such as IBM HTTP Server

You can generate port and URL data from your brokers to use for connecting to a web server such as IBM HTTP Server. You can also generate configuration files for WebSphere Application Server plug-ins or Apache `mod_proxy` modules to use for connecting to web servers. For more information, see Using external web servers with IBM Integration Bus.

Java Architecture for XML Binding (JAXB) support in JavaCompute nodes

You can use Java Architecture for XML Binding (JAXB) with a JavaCompute node to process your messages by accessing, creating, and manipulating JAXB Java object classes that you generate from your message model schema files. For more information, see Using JAXB with a JavaCompute node.

Java shared classloader

Two new shared classloading options are introduced:

- **Execution group classloading** allows only a single defined execution group to access and load any JAR files that are placed in the execution group `shared-classes` directory.
- **Broker-level classloading** allows only a single defined broker to access and load any JAR files that are placed in brokers level `shared-classes` directory.

A new classloading precedence order is also defined.

For more information, see Java shared classloader.

Improvements to Compute nodes

You can now specify a Java classloader in your ESQL to use for loading your Java methods. For more information, see Configuring classloaders for ESQL routines.

Mobile patterns that use IBM Worklight®

You can use the Worklight patterns to integrate your mobile applications with enterprise applications. For more information, see Built-in patterns.

Dynamic operation management

New capabilities facilitate the design of solutions for flexible change with appropriate control.

.NET application domains

You can now create a .NET application domain to package .NET assemblies, and other associated resources, in a BAR file. The .NET application domain is deployed to the run time as a first-class object.

For more information, see .NET application domains overview.

Setting FTP and SFTP servers dynamically

You can override the `Remote server and port` property on the FileOutput node by setting a value in the local environment. You can also use the local environment to specify commands to run before or after an FTP or SFTP transfer finishes.

For more information, see Local environment overrides for the remote server on the FileOutput node .

Deploy message flows as `.msgflow` files

You can now add message flows to broker archive (BAR) files as source `.msgflow` files and deploy these BAR files. The message flows are not compiled into `.cmf` files so if you want to view the message flow source file for a deployed BAR file, you can get this source file directly from the BAR file. For more information, see Broker archive and Adding files to a broker archive.

`mqsipackagebar` command

You can now create broker archive (BAR) files on computers that do not have IBM Integration Bus installed, by using the **`mqsipackagebar`** command. For more information, see Creating a broker archive (BAR) file.

Web user interface security

You can control access to broker resources through the web user interface and the RESTful application programming interface (API). As a broker administrator, you can create web user accounts. The web user accounts have security permissions that are based on their role, which is an associated system account. The permissions are checked to determine the users' authorization to complete tasks in the web user interface or the RESTful application programming interface (API).For more information, see Web and REST security.

Record and replay enhancements

The record and replay capability is enhanced to include support for Oracle databases and role-based security, and enhancements to the web user interface for viewing and replaying data.For more information, see Recording, viewing, and replaying data.

`mqsiwebuseradmin` command

You can use the **`mqsiwebuseradmin`** command to administer user accounts for the web user interface. You can use this command to create or remove a web user, set or change a web user's password, or assign a web user account to a role.For more information, see **`mqsiwebuseradmin`** command.

Extended monitoring support

Support for monitoring events is extended so that all nodes can produce bit streams, which can be included in monitoring events.

Support for `mqsimode` command on z/OS

The **`mqsimode`** command can be run on z/OS by customizing and submitting BIPMODE; see **`mqsimode`** command for more information.

Troubleshooting and support

This fix pack simplifies the collection of broker-related diagnostic information for submission to IBM.

Display the full content of BIP messages

You can view the full content of a runtime BIP message, including the user response and explanation sections, by using the **`mqsiexplain`** command.

For more information, see **`mqsiexplain`** command.

IBM Support Assistant Data Collector
Using IBM Support Assistant Data Collector, which is installed with IBM Integration Bus, you can collect diagnostic documents and submit a problem report to IBM.

For more information, see IBM Support Assistant Data Collector.

Additional problem collector for IBM Support Assistant Data Collector
Use the broker problem collector, which is installed with IBM Support Assistant Data Collector, to gather more extensive broker diagnostic documents.

For more information, see Selecting a problem collector for IBM Support Assistant Data Collector.

Platforms and environments

Additional features that detail changes to operational modes, along with improved database support, and installer enhancements.

Operation Modes
Three of the Operation Modes are renamed, as detailed in the following table:

Previous Name	Current Name
Entry	Express
Starter	Standard
Enterprise	Advanced

Note: The previous names are still supported for migration purposes.

For more information, see "Operation modes" on page 66.

Oracle support for Windows 64-bit
Oracle database support is extended to include Windows 64-bit by means of the ODBC API.

For more information, see Supported databases.

Preemptive database connect
Option to allow database connections to be made before a flow acquires a message, rather than during flow processing. Removes initial connection latency from the message processing.

For more information, see User database connections.

UnixODBC database driver manager improvements
For Linux and UNIX platforms, ODBC connections are now configured by using the unixODBC driver manager for all supported databases. In previous releases, this driver manager was used only for solidDB®. All ODBC definitions are now made in two new configuration files, which are described in Enabling ODBC connections to the databases. The IBM Integration ODBC Database Extender provides the unixODBC driver manager, and is automatically installed after the IBM Integration Bus component is installed.

New installer and uninstaller for the Integration Bus component on Linux, UNIX, and Windows platforms
The new Integration Bus component installation and uninstallation wizard introduces new command-line options for installing and uninstalling. It

also provides the option of completing installation as a user without root authority, and offers a new sample script file for installing IBM Integration Bus components and prerequisite products in silent mode. For more information, see Installing the Integration Bus component and Uninstalling the Integration Bus component.

DataDirect V7.0 driver support
On Windows, Linux and UNIX systems, IBM Integration Bus ships with the DataDirect V7.0 ODBC drivers alongside the default V6.0 ODBC drivers. If required, IBM Integration Bus can be switched to using the DataDirect V7.0 ODBC drivers.

ODBC Activity log
ODBC Activity log provides a high-level overview of how IBM Integration Bus interacts with databases so that you can better understand these interactions.

For more information, see Activity logs.

What else is new if you are migrating from WebSphere Message Broker Version 6.1?

Learn about additional functions in IBM IBM Integration Bus Version 9.0 that are new if you are migrating from WebSphere Message Broker Version 6.1.

The following features and capabilities were introduced in WebSphere Message Broker Version 7.0 and WebSphere Message Broker Version 8.0 and are included in IBM Integration Bus Version 9.0. For information about the features and capabilities that are introduced in IBM Integration Bus Version 9.0, see Chapter 3, “What's new in Version 9.0?,” on page 5.

- “Simplicity and productivity”
- “High performance and scalability” on page 25
- “Application development enhancements” on page 27
- “Deployment and administration enhancements” on page 28
- “Web services enhancements” on page 29
- “Universal connectivity for SOA” on page 29
- “Dynamic operation management” on page 35
- “Troubleshooting and support” on page 38
- “Platforms and environments” on page 38

Simplicity and productivity

The product architecture is simplified, and the product has fewer prerequisite products.

Streamlined components and prerequisite product requirements

Version 9.0 consists of a single runtime component, the broker. All commands and other programs now connect directly to a broker. The broker security model is now implemented by using WebSphere MQ queues, and therefore handles both brokers and queue managers.

Brokers maintain configuration data in the local file system

Brokers create and manage configuration data in an internal repository in the local file system, and have no requirement for a

database. You can back up and restore the broker component and its internal repository by using the commands **mqsibackupbroker** and **mqsirestorebroker**.

Database support for message flows and user data is unchanged; however, supported versions of Relational Database Management Systems (RDBMS) (supplied by IBM and other vendors) have been updated on some platforms.

For more information about database support, see Supported databases.

Optimized deployment and interactions with the broker

Applications that manage brokers and their resources connect directly to the broker. These applications include:

- The IBM Integration Explorer, which is described later in this section.
- The IBM Integration Toolkit.
- Commands; for example, **mqsideploy**, **mqsilist**, and **mqsistartmsgflow**.
- All applications that are written to the Configuration Manager Proxy (CMP) API, (now renamed IBM Integration API).

  For more details about the IBM Integration API, see "The IBM Integration API" on page 72.

You can control which users and applications are able to run commands against particular brokers or execution groups by using WebSphere MQ security, as described in the following section.

Administration security

Set up broker administration security to control the authority that is required by users to complete specific administrative tasks. You can enable security when you create a broker, or change it later on an existing broker. This option, which uses WebSphere MQ facilities, replaces Access Control Lists (ACLs) that were managed by the Configuration Manager in previous versions.

For further details, see Administration security overview.

Policy set bindings editor enhancements

Use the **mustUnderstand** attribute in the Policy Sets and Policy Set Bindings editor to configure the security header of the consumer message. For more information, see Policy Sets and Policy Set Bindings editor: Advanced panel

For details of new and updated commands, see Commands.

IBM Integration Explorer

The IBM Integration Explorer is an administration interface that is integrated as a plug-in into WebSphere MQ Explorer, so that you can administer both brokers and WebSphere MQ queue managers on local and remote computers.

Important status information is always on view, and you can access details about what each broker is doing, and has recently done. Configuration and other changes are monitored, and the user responsible for these changes is recorded.

The IBM Integration Explorer supports all the function that is provided by the Broker Administration perspective in the WebSphere Message Broker Toolkit in previous versions, and offers additional, more advanced features. Limited administration functionality is still available in the new Integration Nodes view in the IBM Integration Toolkit to run a subset of operations.

The IBM Integration Explorer includes the following capabilities:

- You can create, delete, start, and stop local brokers without using the command line.
- You can see the relationships between your brokers and your queue managers.
- You can deploy a broker archive file to multiple execution groups in one step.
- You can see visualizations of your accounting and statistics data.
- You can see visualizations of your resource statistics data.
- You can configure broker properties, including creating and modifying configurable services.
- You can view the broker administration queue, and remove pending tasks that have been submitted to the broker.
- You can connect and configure settings for a DataPower® device.

For further details about the IBM Integration Explorer, see "IBM Integration Explorer" on page 73.

For further details about the Integration Nodes view in the IBM Integration Toolkit, see Integration Nodes view.

Publish/subscribe support

All topic-based publish/subscribe operations are handled by WebSphere MQ. You can use IBM Integration Bus facilities to extend publish/subscribe options to include content-based publishers and subscribers.

All applications use a single topic space that is managed by WebSphere MQ, and access control is handled by the queue manager. The concept of broker domains, which is valid in previous versions of IBM Integration Bus, no longer exists; equivalent function for broker domains, broker topologies, and broker collectives is provided by WebSphere MQ clusters.

The Publication node uses WebSphere MQ publish/subscribe facilities. You can use the *NoMatch* terminal to identify scenarios in which no subscribers are registered to receive particular topics.

With these changes, your publish/subscribe network requires you to configure a broker only where you have content-based subscribers, not throughout the network. The content-based filters that you can specify can now include full ESQL expressions, including namespace support.

You can migrate JMS applications, and applications that use the MQRFH2 header, directly to WebSphere MQ.

WebSphere MQ Real-time Transport and WebSphere MQ Telemetry Transport nodes are no longer supported. Therefore, the following nodes have been removed:

- Real-timeInput
- Real-timeOptimizedFlow
- SCADAInput
- SCADAOutput

If you use the WebSphere MQ Real-time Transport, your applications can use equivalent qualities of service provided by WebSphere MQ. Migrate JMS real-time publishers and subscribers to the "read-ahead get" and "asynchronous-put" facilities of WebSphere MQ.

Message flows containing these nodes that you have migrated to IBM Integration Bus Version 9.0 will not start until these nodes have been removed from the flow, and the flow has been redeployed.

Contact your account representative for more information about support for WebSphere MQ Real-time Transport and WebSphere MQ Telemetry Transport.

For more information about the changes to publish/subscribe see Routing using publish/subscribe applications.

IBM Integration Toolkit

You can administer local and remote brokers in the Integration Nodes view. This view is integrated into the Integration Development perspective, so that you can access basic administration tasks while you are developing, deploying, debugging, and testing your applications.

Samples are now accessible through the Samples page, and from the information center in the IBM Integration Toolkit. For a full list of the samples, see Chapter 7, "Samples," on page 87.

Patterns

A pattern is a reusable solution that encapsulates a top-down tested approach to solving a common architecture, design, or deployment task in a particular context. This approach complements bottom-up development of creating message flows and nodes.

A number of patterns are supplied in the IBM Integration Toolkit, and you can use the Patterns Explorer, which includes comprehensive help, to simplify creation of common scenarios.

You can configure these patterns with values for use in your own environment to solve specific business problems. The supplied patterns use preferred techniques in message flow design, to produce efficient and reliable flows.

For more information, see Patterns.

User-defined patterns

User-defined patterns extend the function of IBM Integration Bus so that you can create patterns that you can reuse within your organization.

For more information, see User-defined patterns.

Modifying pattern instances by using Java or PHP

Use Java or PHP code to modify pattern instances when the pattern user generates an instance of a user-defined pattern. For example, to modify the structure of a message flow based on the values of pattern parameters.

For more information, see Modifying pattern instances by using Java or PHP.

Share your user-defined patterns with other users

Package your user-defined pattern into a pattern archive so that the user-defined pattern can be distributed to pattern users by adding the pattern archive to a pattern community site.

For more information, see Packaging and distributing pattern plug-ins.

Using a subflow as a user-defined node

Develop a user-defined node that packages a subflow, either in the same way that you create any other user-defined node that has its implementation based on Java, or by basing it on an existing subflow.

For more information, see Using a subflow as a user-defined node.

WebSphere MQ Telemetry Transport

WebSphere MQ Telemetry Transport is supported from WebSphere MQ. For more information, see Changes to nodes in IBM Integration Bus Version 9.0.

Web administration

You can use the IBM Integration Bus web user interface to administer broker resources. The web user interface enables web users to access broker resources through an HTTP client, and provides broker administrators with an alternative to the IBM Integration Explorer for administering broker resources. The web user interface listener is enabled by default for new brokers.

For more information, see Administering brokers using the web user interface.

RESTful API support

IBM Integration Bus supports the REST management API for broker administration.

For more information, see Representational State Transfer (REST) API.

Converting projects to applications and libraries

If you imported resources from a previous version of WebSphere Message Broker, you can convert all or some of your projects automatically to applications and libraries. A refactoring wizard applies rules to decide how to convert your resources. The wizard identifies errors that would prevent successful conversion, and provides fixes for those errors, where possible.

For more information, see Converting a project to an application or library.

Service creation

Web services are now a first-class artifact in IBM Integration Bus. You can create web services in the new Services editor, and implement the operations as message flows.

For more information, see Developing integration solutions by using integration services.

Export mapping information

You can now export mapping information from Compute nodes, for use in impact analysis and data lineage operations in products such as IBM InfoSphere Metadata Workbench. See Exporting mapping information from Compute nodes.

New `jdbcProviderXASupport` property

An optional property that controls whether the broker connects to a database server by using XA Protocol. For more information, see Setting up a JDBC provider for type 4 connections.

Creating multi-instance brokers with no domain controller restrictions

You can now create a multi-instance broker without installing Windows on a domain controller. You can also create a multi-instance broker and a multi-instance message queue at the same time.

For more information, see Creating a multi-instance broker and Creating a multi-instance broker and a multi-instance queue manager .

Analyze and filter information in XML files

The Data Analysis perspective analyzes and filters information in XML files. You can use this analysis to create Data Analysis tools to transform your data.

For more information, see Data Analysis.

Handling large messages in Java

When you design a message flow that handles large messages that are made up of repeating structures, you can code Java methods that help to reduce the storage load on the broker.

For more information, see Working with large input messages to propagate multiple output messages.

High performance and scalability

The use of a wide range of hardware, software, and virtualized environments is facilitated.

WebSphere eXtreme Scale global cache

A global cache is a repository for data that you want to reuse. The cache facilitates sharing of data across processes, and eliminates the need for an alternative solution, such as a database. The global cache is embedded in the broker. The cache has a default topology and can be used immediately without any configuration. However, you can turn off the default configuration and set properties explicitly for each execution group.

For more information, see Data caching overview.

Developer Edition

IBM Integration Bus Developer Edition is a full function version of the product, which you can use for evaluative purposes. You can download the Developer Edition at no charge and you are free to use it for as long as you require, within the terms of the license.

Developer mode is introduced with the Developer Edition. All brokers that you create in the Developer Edition are created in Developer mode by default.

There is no longer a Trial Edition for IBM Integration Bus.

Developer Edition is restricted to processing one message per second.

For more information, see “Operation modes” on page 66.

External WebSphere eXtreme Scale grids

An embedded global cache was provided in the previous version as a repository for data that you want to reuse. In addition to the grid that is available (as the embedded global cache) in IBM Integration Bus, you can

now integrate with WebSphere eXtreme Scale grids that are running elsewhere. You can work with multiple external grids, and the embedded grid, at the same time. You can also enable SSL for connections to external WebSphere eXtreme Scale grids.

Enable the new capability by using the **-f** parameter on the **mqsichangebroker** command, as described in **mqsichangebroker** command. You must stop the cache before you can enable the capability for the brokers.

For more information, see WebSphere eXtreme Scale grids.

Using a domain name to identify grids

WebSphere eXtreme Scale clients use a domain name to identify and distinguish between embedded grids. Only WebSphere eXtreme Scale servers with the same domain name can participate in the same grid. If you do not specify a domain name, the broker creates a name that is based on the server names of the catalog servers.

By default, each server starts with a domain name that is derived by the broker. In previous versions of WebSphere Message Broker, the domain name for all servers in all embedded caches was an empty string. Servers in different domains cannot collaborate in the same grid. Therefore, for a cache that spans more than one broker, you must enable the new capability for these brokers at the same time.

Brokers on IBM Integration Bus Version 9.0 can still collaborate in the same grid as brokers on WebSphere Message Broker Version 8.0.0.2 or earlier. In this scenario, the capability should not be enabled on the WebSphere eXtreme Scale servers, and then the servers continue to use an empty string as the domain name; servers report a BIP7140 warning message to the system log with this information.

Enable the capability by using the **-f** parameter on the **mqsichangebroker** command, as described in **mqsichangebroker** command.

For more information, see Configuring the embedded global cache by using commands.

Removing data from the global cache

You can specify how long data exists in the global cache by setting a *time to live* value. You specify this value when you get an MbGlobalMap object. The value applies to all cache entries that are created by using that MbGlobalMap object in that instance of the JavaCompute node. After the specified time, the affected data is removed from the global cache automatically.

Enable the new capability by using the **-f** parameter on the **mqsichangebroker** command, as described in **mqsichangebroker** command. You must stop the cache before you can enable the capability for the brokers.

For more information, see Embedded global cache.

Scale mode

Scale mode provides support for unlimited execution groups and a defined subset of nodes. In Scale mode, you can create multiple brokers and deploy message flows containing the supported nodes to your execution

groups. WebSphere Enterprise Service Bus customers can obtain a transfer license, which enables them to migrate to IBM Integration Bus and to create and run brokers in Scale mode.

For more information, see "Operation modes" on page 66.

New `timeoutThreads` property

An optional property that assigns additional processing threads to enable processing of timed out aggregation messages in the AggregateReply node. For more information, see Processing timed out aggregation messages.

Application development enhancements

Application development is simplified.

Resource statistics for JMS
View the JMS statistics to view the number of JMS connections that are used and messages that are processed by nodes that use JMS transport. For more information, see Resource statistics.

Activity Logs
Use the new Activity logs to get an overview of recent activities in your message flows and associated external resources.

For more information, see Using Activity logs.

Deploying flows and applications in stopped state
You can specify how a message flow or application is started after it is deployed, or after the broker, execution group, or containing application is restarted. You can choose to start an application or flow manually, or for it to be started automatically. You can also choose to maintain the existing state of an application or library.

For more information, see Configuring the start mode of flows and applications at development time.

Deployable subflows
Create subflows that you can deploy to a broker as individual resources. Use the deployable subflow in more than one message flow application so that when you change the subflow and redeploy it, all your message flow applications use the updated subflow.

For more information, see Subflows.

Deployable ESQL
Create ESQL files that you can deploy to a broker as individual resources. Use the code from the deployed ESQL file in more than one message flow application so that when you change the ESQL file and redeploy it, all your message flow applications that reference the ESQL code are updated.

For more information, see Deploying an ESQL file.

Dynamic configurable services
Your changes to most types of configurable service will take effect from the next time a message flow that uses the configurable service is called. For some types of configurable service, for which it is explicitly stated in the documentation, you must stop and start the execution group for the change to take effect.

For more information, see Configurable services.

Record and replay

For audit purposes or problem determination, you might want to keep a record of messages that pass through a message flow. You can record those messages in a database, then view them. For more information, see Recording, viewing, and replaying data.

Deployment and administration enhancements

Additional features provide better information and control of operations.

Resource statistics for JMS
View the JMS statistics to view the number of JMS connections that are used and messages that are processed by nodes that use JMS transport. For more information, see Resource statistics.

Activity Logs
Use the new Activity logs to get an overview of recent activities in your message flows and associated external resources.

For more information, see Using Activity logs.

Deploying flows and applications in stopped state
You can specify how a message flow or application is started after it is deployed, or after the broker, execution group, or containing application is restarted. You can choose to start an application or flow manually, or for it to be started automatically. You can also choose to maintain the existing state of an application or library.

For more information, see Configuring the start mode of flows and applications at development time.

Deployable subflows
Create subflows that you can deploy to a broker as individual resources. Use the deployable subflow in more than one message flow application so that when you change the subflow and redeploy it, all your message flow applications use the updated subflow.

For more information, see Subflows.

Deployable ESQL
Create ESQL files that you can deploy to a broker as individual resources. Use the code from the deployed ESQL file in more than one message flow application so that when you change the ESQL file and redeploy it, all your message flow applications that reference the ESQL code are updated.

For more information, see Deploying an ESQL file.

Dynamic configurable services
Your changes to most types of configurable service will take effect from the next time a message flow that uses the configurable service is called. For some types of configurable service, for which it is explicitly stated in the documentation, you must stop and start the execution group for the change to take effect.

For more information, see Configurable services.

Record and replay

For audit purposes or problem determination, you might want to keep a record of messages that pass through a message flow. You can record those messages in a database, then view them. For more information, see Recording, viewing, and replaying data.

Web services enhancements

Web services enhancements improve security and reliability.

Web Services Reliable Messaging
Administrators can configure message flows and nodes to use WS-RM (Web Services Reliable Messaging) for inbound and outbound SOAP messages. For more information, see Web Services Reliable Messaging.

Internationalized Domain Names (IDNs)
Non-English host names can be used in web browsers to accommodate non-native English speakers as a significant group of Internet users.

SSL client authentication selection key
SSL-based nodes can specify a key alias for use by the JSSE layer for a key in the keystore to be used for a specified connection.

Universal connectivity for SOA

Additional nodes and configurable services expand the interaction of the broker with other products.

Support for IBM Sterling Connect:Direct® file transfer

IBM Sterling Connect:Direct is a managed file transfer product that transfers files between, and within, enterprises.

Two new nodes have been added to implement the additional features. These nodes are the CDInput and CDOutput nodes. For further information about these nodes, see CDInput node and CDOutput node.

Note, that IBM Integration Bus nodes work purely as clients connecting to the external Connect:Direct server, using the IBM Sterling Connect:Direct Java Application Interface.

For an overview of IBM Sterling Connect:Direct, see IBM Sterling Connect:Direct overview and concepts

SCA nodes for WebSphere Process Server

Five built-in message flow nodes are provided to improve the interaction between IBM Integration Bus and WebSphere Process Server Version 6.2 by using Web Services (SOAP over HTTP) or WebSphere MQ bindings.

The nodes are the SCAInput, SCAReply, SCARequest, SCAAsyncRequest, and SCAAsyncResponse nodes.

For more information, see Service Component Architecture (SCA) overview.

Enhanced support for the PHPCompute node

Support for the PHP scripting language is available on all operating systems on which IBM Integration Bus is supported, except Solaris on x86-64. The PHPCompute node supports general-purpose transformation logic in the PHP language, and complements the Compute, JavaCompute, XSLTransform, and Mapping nodes. In addition, the set of supported PHP extensions has been increased. For more information, see Using PHP and PHP extensions.

SAP, Siebel, and Peoplesoft enhancements

Improved connectivity with Enterprise Information Systems.

New WebSphere Adapter nodes

Use the SAPReply node to send a reply to an SAP synchronous callout. Use this node with an SAPInput node to implement a message flow application that acts as a remote function call (RFC) destination.

For more information, see SAPReply node and BAPI inbound scenarios.

Generic IDoc routing

By using the SAPInput node in passthrough mode, IBM Integration Bus can receive any IDoc, and route it according to IDoc type. By using this method, you can also use a single RFC program ID to receive all IDoc types, while still allowing individual IDoc processing.

For more information, see Generic IDoc routing.

SAP high availability

You can deploy an SAP adapter and a message flow that contains an SAPInput node to two brokers on your network; these brokers can accept IDocs concurrently from the same SAP system so that you can build a highly available environment.

On distributed systems, two brokers share state by using queues on a third queue manager, which is running in multi-instance mode. Each broker has client connections to that queue manager.

On z/OS, the shared state is stored on a shared queue. Each broker connects to the queue sharing group.

For more information, see SAP high availability.

Iterative discovery

You can take an adapter component that was created by using the Adapter Connection wizard in IBM Integration Bus, and update it with newly discovered objects from the Enterprise Information System (EIS) by running the Adapter Connection - Iterative Discovery wizard. This facility is known as *iterative discovery*. You can either add the new objects without modifying existing objects, or replace existing objects.

For more information, see Enhancing existing adapters with newly discovered objects.

Iterative deployment

If your message flow acts as a gateway to an EIS, you can use it to call new services that did not exist when you developed the flow. You can also create an event handler to an EIS to handle new event types that did not exist when you first developed your message flow. In both cases, if a new service or event is provided by the EIS, you do not have to modify and retest the message flow. This facility is called *iterative deployment*.

For more information, see WebSphere Adapters deployment.

CICSRequest node enhancements

- You can specify a mirror transaction name on the CICSRequest node, which you can use to run CICS Transaction Server for z/OS tasks and

programs. This grouping greatly assists in collecting statistics, accounting, and aids decision making about task priority. For more information about mirror transactions, see CICS Transaction Server for z/OS mirror transactions.

- The CICSRequest node support in IBM Integration Bus provides direct communication with CICS (two-tier connection) by sending Distributed Program Link (DPL) requests over TCP/IP-based IP InterCommunications (IPIC) protocol, or communication with CICS through CICS Transaction Gateway for Multiplatforms (three-tier connection). For more information about the two-tier and three-tier connection models, see CICS Transaction Server for z/OS overview for a high-level overview, or CICS Transaction Server for z/OS two-tier connectivity and CICS Transaction Server for z/OS three-tier connectivity for detailed conceptual information.
- You can specify either a COMMAREA data structure or a channel data structure on the CICSRequest node to use as input for linking to CICS programs. The data structure that is specified as input returns the same data structure as output. Channels are an alternative for COMMAREAs, providing relief from the COMMAREA maximum size of 32766 bytes, and allowing greater flexibility in input and output data structures. For more information about using a COMMAREA or channel data structure, see COMMAREA or channel data structures.
- The CICS Transaction Server for z/OS Channel Connectivity sample demonstrates how to call a channel-based CICS program. A CICS channel structure can be represented in IBM Integration Bus by a message collection. This sample demonstrates how to create and populate a message collection for the CICSRequest node and how to process the collection after the call.

Sequence and Resequence nodes

IBM Integration Bus provides support for adding sequence numbers to messages, and for reordering messages in the message flow based on their sequence number. You can use the new Sequence node to add sequence numbers to the messages, and the new Resequence node to reorder the messages into their original sequential order.

For more information, see Sequence node and Resequence node.

New EmailInput node

Use the EmailInput node to retrieve an email, with or without attachments, from an email server that supports Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP).

For more information, see EmailInput node.

A sample that demonstrates how you can use the EmailInput node is also provided. For more information, see Email.

New JDEdwardsInput and JDEdwardsRequest nodes

Use the JDEdwardsInput and JDEdwardsRequest nodes to interact with a JD Edwards EnterpriseOne server. For example, you can use the JDEdwardsRequest node to discover JD Edwards EnterpriseOne business functions, XML lists, and real-time events.

For more information, see JDEdwardsInput node and JDEdwardsRequest node.

A sample that demonstrates how you can use the JDEdwardsRequest node node is also provided. For more information, see JD Edwards Connectivity.

New FileRead node

Use the FileRead node to read one record, or the entire contents of a file, from within a message flow.

For more information, see FileRead node.

The Message Routing sample has been enhanced, and now demonstrates how to use the FileRead node. For more information about how to process messages based on the contents of an XML or CSV file, see Message Routing.

FileInput node

Skip the first record in a file. The FileInput node reads the first record in the file but does not propagate the record to the Out terminal. Records are propagated as normal, from the second record onwards. Use this option when the first record is a header that does not need to be processed. It is not valid to use this option when using the whole file.

For more information, see FileInput node.

SSL for TCP/IP nodes

You can configure the broker to use SSL for TCP/IP connections; see Configuring TCP/IP client nodes to use SSL.

HTTP compression for HTTP and SOAP nodes

You can configure HTTP nodes to use HTTP compression and decompression when sending and receiving messages. You can also configure SOAP nodes to use HTTP compression functionality. For more information, see Using compression with HTTP and SOAP nodes.

HTTP property hostnameChecking

Use the HTTP Transport property hostnameChecking to specify whether the host name of the server that is receiving the request must match the host name in the SSL certificate. For more information, see HTTPRequest node, and SOAPRequest node.

JMS transport for SOAP nodes

The SOAPAsyncRequest and SOAPAsyncResponse nodes support JMS as well as HTTP transport. You can import WSDL with bindings for both JMS and HTTP transport, and switch transports for the SOAPAsyncRequest node during a message flow. WS-Security and WS-Addressing are supported for SOAP/JMS, as well as transactionality. IBM Integration Bus supports both W3C (standard) and IBM (proprietary) WSDL formats for SOAP/JMS. For more information, see WSDL URI formats for JMS and SOAP over JMS.

Improved *order by* **support for the MQInput node**

You can now sort by any element in the message. For each value of that element, the messages are processed in arrival order. See Optimizing message flow throughput and MQInput node.

Web services gateway

The SOAP nodes support a web services gateway mode, which does not require a WSDL to configure the SOAP nodes, and allows IBM Integration

Bus to handle generic SOAP request/response and one-way messages when used as a web services provider or consumer. IBM Integration Bus can also act as a façade between multiple web service clients and multiple back-end web service providers.

For more information, see Gateway operation mode for SOAP nodes.

A sample that demonstrates how you can use a web services gateway is also provided. For more information, see Web Services Gateway.

DatabaseInput node

The DatabaseInput node can now generate the code for a simple database query. After code generation, you can add custom code; see Configuring a DatabaseInput node. By using an MQInput node with the built-in DatabaseInput node the WBI JDBC Adapter Migration sample re-creates a scenario of migrating a JDBC adapter to invoke a message flow.

A sample that demonstrates how you can use the DatabaseInput node is also provided. For more information, see DatabaseInput Node.

New configurable services for EDA nodes

You can use the following configurable services to define the WebSphere MQ queues on which EDA nodes store event state:

- Aggregation
- Collector
- Resequence
- Timer

You can also use these configurable services to specify timeouts for the nodes. For more information, see Configurable services properties.

New configurable service properties

- Use the JDEdwardsConnection configurable service property assuredOnceDelivery to specify whether to provide assured once-only delivery for inbound events. For more information, see Configurable services properties.
- Use the MonitoringProfiles configurable service property useParserNameInMonitoringPayload to determine when the payload is included in a monitoring message. For more information, see Configurable services properties and Configuring monitoring event sources using a monitoring profile.
- Use the IMSConnect configurable service property CodedCharSetID to change your IMS™ system or IMSConnect CCSID from the default value. For more information, see Changing connection information for the IMSRequest node and Configurable services properties.
- Use the FtpServer configurable service property preserveRemoteFileDate to specify whether files that are retrieved from a remote server by the FileInput node retain the last modified date on the server. For more information, see FtpServer configurable service and Configurable services properties.

JSON domain

Fix pack v7.0.0.2 provides support for a JSON domain. Messages in the JSON domain are processed by the JSON parser and serializer. The JSON parser interprets a bit stream by using the JSON grammar, and generates a corresponding JSON domain logical message tree in the broker.

For more information, see JSON parser and domain.

RESTful Web Service Using JSON sample

This sample shows how you can use IBM Integration Bus to front an existing service as a RESTful web service, providing a JSON message format interface. The sample also shows how to consume the RESTful Web Service from a message flow.

For more information, see RESTful Web Service Using JSON.

Web Service Aggregation sample

The sample demonstrates how you can invoke a number of web services and amalgamate the results by using IBM Integration Bus aggregation nodes. The sample illustrates how you can use aggregation for transports other than WebSphere MQ, and highlights any issues of which to be aware. For more information, see Web Service Aggregation.

TDS mnemonics enhancement

Use the TDS mnemonic string <X12_ERS> as an element repetition separator for X12. For more information, see Message Sets: TDS Mnemonics.

JMSReceive node

Use the new JMSReceive node to consume or browse messages from a JMS queue in the middle of a message flow.

For more information, see JMSReceive node.

Improved integration with WebSphere Application Server

You can view your broker resources, including web services, by using IBM Integration Bus administration for WebSphere Application Server. See Administering integration nodes from WebSphere Application Server.

DFDL improvements

You can import C header files; see Creating a DFDL schema file by using the New Message Model wizard. Field length prefixes are supported. A new industry sample is provided, consisting of a DFDL schema for ISO8583, and a message flow for transforming ISO8583 messages to XML, and vice versa.

Graphical Data Mapping Enhancements

You can modify database content from your Graphical Data Maps; see Modifying data in a database by using mapping .

You can also convert a message map from a previous version of IBM Integration Bus to a graphical data map; see Converting a message map from a `.msgmap` file to a `.map` file.

SOAP nodes can use the broker-wide HTTP listener

You can configure your execution groups so that the SOAP nodes use the broker-wide HTTP listener to process HTTP messages, rather than the execution group listener. Using the broker-wide listener can simplify the administration of your broker configuration. For more information, see HTTP listeners.

HTTP asynchronous request-response

Use new HTTP asynchronous nodes to call an HTTP web service and receive an asynchronous response without blocked waiting. For more information, see Using HTTP asynchronous request-response.

You can also use HTTP asynchronous response-request behavior instead of WS-Addressing with the SOAPAsyncRequest node to make HTTP requests and receive an asynchronous response. For more information, see Choosing asynchronous behavior for the SOAPAsyncRequest node.

Enhanced support for external web servers such as IBM HTTP Server

You can generate port and URL data from your brokers to use for connecting to a web server such as IBM HTTP Server. You can also generate configuration files for WebSphere Application Server plug-ins or Apache `mod_proxy` modules to use for connecting to web servers. For more information, see Using external web servers with IBM Integration Bus.

Java Architecture for XML Binding (JAXB) support in JavaCompute nodes

You can use Java Architecture for XML Binding (JAXB) with a JavaCompute node to process your messages by accessing, creating, and manipulating JAXB Java object classes that you generate from your message model schema files. For more information, see Using JAXB with a JavaCompute node.

Java shared classloader

Two new shared classloading options are introduced:

- **Execution group classloading** allows only a single defined execution group to access and load any JAR files that are placed in the execution group `shared-classes` directory.
- **Broker-level classloading** allows only a single defined broker to access and load any JAR files that are placed in brokers level `shared-classes` directory.

A new classloading precedence order is also defined.

For more information, see Java shared classloader.

Improvements to Compute nodes

You can now specify a Java classloader in your ESQL to use for loading your Java methods. For more information, see Configuring classloaders for ESQL routines.

Mobile patterns that use IBM Worklight

You can use the Worklight patterns to integrate your mobile applications with enterprise applications. For more information, see Built-in patterns.

Dynamic operation management

Additional features provide better information and control of operations.

Multi-instance brokers

IBM Integration Bus builds on the multi-instance queue manager support introduced in WebSphere MQ Version 7.0.1 to provide a highly available configuration with active and passive brokers.

Multi-instance brokers and queue managers store their configurations on shared network storage so that if a failure occurs in an active component, the passive component assumes the configuration and operation of the

active component. The use of queue managers in this way avoids the requirement for a high availability solution, such as HACMP™, supplied by a vendor software company.

For further information, see Configuring for high availability.

Audit and monitoring

You can now generate comprehensive audit and monitoring events from message flows, either at design time or operationally, for new and existing message flows. These events can be consumed by a diverse range of applications and systems, including WebSphere Business Monitor, WebSphere MQ and JMS applications, and vendor applications. In addition to business monitoring, you can use these events for business intelligence, and audit scenarios.

See Business-level monitoring for an overview of monitoring.

The following improvements to the monitoring of message flows are introduced:

- A filter can be applied to every event source, to control whether the event is emitted. See Configuring monitoring event sources using monitoring properties and Monitoring profile.
- You can export the monitoring information about a message flow from the IBM Integration Toolkit, and import it into WebSphere Business Monitor Version 7.0 to generate a monitor model for your message flow. See Creating a monitor model for WebSphere Business Monitor V7 or later.
- You can choose whether the emission of monitoring events by a message flow is coordinated with the message flow transaction, is in an independent unit of work, or is not in a unit of work, which improves overall monitoring performance; see Monitoring basics.
- Monitoring events now contain an integer counter, as well as the creation time of the events, for use in sequencing events. The Sequence tab has been removed from the IBM Integration Toolkit.

Resource statistics

You can collect statistics for some of the resources that are used by execution groups in the broker to help with problem diagnosis and broker optimization. Supported resources are the Java Virtual Machine (JVM), and the outbound sockets. For example, you can monitor the sockets that are used by SOAP nodes in your message flows.

You can start and stop statistics collection at broker or execution group level by using the IBM Integration Explorer, the Integration API, or the **mqsichangeresourcestats** command.

The resource statistics framework is based on the existing accounting and statistics for message flows, and generates periodic messages as publications that your programs can subscribe to. You can also view these statistics in the IBM Integration Explorer, which provides both numeric and graphical representations.

For further details, see Analyzing resource performance.

Resource statistics for parsers

View the statistics for a parser to view the number of input and output

messages processed by a message flow, and determine if message flow parsers are using large amounts of memory. For more information, see Resource statistics.

.NET application domains

You can now create a .NET application domain to package .NET assemblies, and other associated resources, in a BAR file. The .NET application domain is deployed to the run time as a first-class object.

For more information, see .NET application domains overview.

Setting FTP and SFTP servers dynamically

You can override the `Remote server and port` property on the FileOutput node by setting a value in the local environment. You can also use the local environment to specify commands to run before or after an FTP or SFTP transfer finishes.

For more information, see Local environment overrides for the remote server on the FileOutput node .

Deploy message flows as `.msgflow` files

You can now add message flows to broker archive (BAR) files as source `.msgflow` files and deploy these BAR files. The message flows are not compiled into `.cmf` files so if you want to view the message flow source file for a deployed BAR file, you can get this source file directly from the BAR file. For more information, see Broker archive and Adding files to a broker archive.

`mqsipackagebar` command

You can now create broker archive (BAR) files on computers that do not have IBM Integration Bus installed, by using the **`mqsipackagebar`** command. For more information, see Creating a broker archive (BAR) file.

Web user interface security

You can control access to broker resources through the web user interface and the RESTful application programming interface (API). As a broker administrator, you can create web user accounts. The web user accounts have security permissions that are based on their role, which is an associated system account. The permissions are checked to determine the users' authorization to complete tasks in the web user interface or the RESTful application programming interface (API).For more information, see Web and REST security.

Record and replay enhancements

The record and replay capability is enhanced to include support for Oracle databases and role-based security, and enhancements to the web user interface for viewing and replaying data.For more information, see Recording, viewing, and replaying data.

`mqsiwebuseradmin` command

You can use the **`mqsiwebuseradmin`** command to administer user accounts for the web user interface. You can use this command to create or remove a web user, set or change a web user's password, or assign a web user account to a role.For more information, see **`mqsiwebuseradmin`** command.

Extended monitoring support

Support for monitoring events is extended so that all nodes can produce bit streams, which can be included in monitoring events.

Support for `mqsimode` command on z/OS

The **`mqsimode`** command can be run on z/OS by customizing and submitting BIPMODE; see **`mqsimode`** command for more information.

Troubleshooting and support

This fix pack simplifies the collection of broker-related diagnostic information for submission to IBM.

Display the full content of BIP messages

You can view the full content of a runtime BIP message, including the user response and explanation sections, by using the **`mqsiexplain`** command.

For more information, see **`mqsiexplain`** command.

IBM Support Assistant Data Collector

Using IBM Support Assistant Data Collector, which is installed with IBM Integration Bus, you can collect diagnostic documents and submit a problem report to IBM.

For more information, see IBM Support Assistant Data Collector.

Additional problem collector for IBM Support Assistant Data Collector

Use the broker problem collector, which is installed with IBM Support Assistant Data Collector, to gather more extensive broker diagnostic documents.

For more information, see Selecting a problem collector for IBM Support Assistant Data Collector.

Platforms and environments

Additional features that detail changes to operational modes, along with improved database support, and installer enhancements.

Operation Modes

Three of the Operation Modes are renamed, as detailed in the following table:

Previous Name	Current Name
Entry	Express
Starter	Standard
Enterprise	Advanced

Note: The previous names are still supported for migration purposes.

For more information, see “Operation modes” on page 66.

Oracle support for Windows 64-bit

Oracle database support is extended to include Windows 64-bit by means of the ODBC API.

For more information, see Supported databases.

Preemptive database connect

Option to allow database connections to be made before a flow acquires a message, rather than during flow processing. Removes initial connection latency from the message processing.

For more information, see User database connections.

UnixODBC database driver manager improvements

For Linux and UNIX platforms, ODBC connections are now configured by using the unixODBC driver manager for all supported databases. In previous releases, this driver manager was used only for solidDB. All ODBC definitions are now made in two new configuration files, which are described in Enabling ODBC connections to the databases. The IBM Integration ODBC Database Extender provides the unixODBC driver manager, and is automatically installed after the IBM Integration Bus component is installed.

New installer and uninstaller for the Integration Bus component on Linux, UNIX, and Windows platforms

The new Integration Bus component installation and uninstallation wizard introduces new command-line options for installing and uninstalling. It also provides the option of completing installation as a user without root authority, and offers a new sample script file for installing IBM Integration Bus components and prerequisite products in silent mode. For more information, see Installing the Integration Bus component and Uninstalling the Integration Bus component.

DataDirect V7.0 driver support

On Windows, Linux and UNIX systems, IBM Integration Bus ships with the DataDirect V7.0 ODBC drivers alongside the default V6.0 ODBC drivers. If required, IBM Integration Bus can be switched to using the DataDirect V7.0 ODBC drivers.

ODBC Activity log

ODBC Activity log provides a high-level overview of how IBM Integration Bus interacts with databases so that you can better understand these interactions.

For more information, see Activity logs.

What's new in IBM Integration Bus for WebSphere Enterprise Service Bus users?

IBM Integration Bus is a compatible evolution of WebSphere Message Broker that is designed to incorporate features that are found in WebSphere Enterprise Service Bus. IBM Integration Bus provides a universal integration capability that addresses a wide range of integration scenarios. These scenarios include web services such as SOAP and REST, messaging, database, file, ERP systems, mobile, physical devices, email, custom systems and more.

IBM Integration Bus offers WebSphere Enterprise Service Bus users additional capabilities in the following areas:

- "Message modeling and message processing" on page 40
- "Patterns framework" on page 42
- "Operational management" on page 43
- ".NET" on page 44
- "File processing" on page 45
- "WebSphere ESB conversion tool" on page 45

To learn about the differences between IBM Integration Bus and WebSphere Enterprise Service Bus, read the following sections:

- "Application development differences" on page 45

- "Terminology differences" on page 48
- "Runtime environment differences" on page 48

Message modeling and message processing

IBM Integration Bus provides functionality to model and process a wider range of message data more efficiently. By using this functionality, you eliminate the need to write custom Java code to parse data structures, you reuse common transformation parsing structures by importing or including them instead of coding, and you accelerate the testing of your integration solutions.

- IBM Integration Bus supplies more built-in parsers than WebSphere Enterprise Service Bus. For a complete list of body message built-in parsers and header message parsers built-in parsers, see Parsers.
- IBM Integration Bus supports the ACORD AL3, CSV, EDIFACT, FIX, HL7, SWIFT, TLOG, X12 and ISO8583 standards. For more information, see Industry standard formats.
- In IBM Integration Bus, you can use the Data Format Description Language (DFDL) to define the structure of general text and binary formatted data in a way that is independent of the data format. DFDL is a standard of the Open Grid Forum (OGF) that enables powerful data interchange and very high-performance data handling. For more information, see Data Format Description Language (DFDL).
- IBM Integration Bus provides a number of mechanisms to model your message data efficiently. For example, you can use an importer to read metadata such as C header files, COBOL copybooks, and EIS metadata, and to create message models from that metadata, or you can create a message model for your SAP data by using DFDL. For more information, see Modeling different data formats.
- IBM Integration Bus provides predefined message models for common industry standard message formats such as SWIFT, EDIFACT, X12, FIX, HL7, and TLOG. For more information, see Modeling different data formats.
- In IBM Integration Bus, you can reuse one message model schema file or message definition file by using one of the two mechanisms that XML schema provides to reuse message definition files: import and include. For more information, see Reusing message model files .
- IBM Integration Bus supports lazy parsing for a very wide range of formats, for example all message formats supported by DFDL. This reduces both the amount of processing performed and the memory consumption for a wide range of integration scenarios. WebSphere Enterprise Service Bus only supports lazy parsing for XML data.

Message modeling and message processing overview

Business data flows between service providers and service consumers over various protocols (HTTP, JMS, WebSphere MQ, EIS, and so on) in various data formats such as comma separated value, delimited, fixed width, and COBOL . Different protocols have different mechanisms for carrying the business data in their protocol envelope. While the transport protocol envelope is different, the business data may or may not be in same format across these protocols.

IBM Integration Bus supplies a range of parsers to parse and write message formats, for example XMLNSC, DFDL, or DataObject:

- The XMLNSC parser is a flexible, general-purpose XML parser that offers high performance XML parsing and optional XML schema validation. The XMLNSC parser has a range of options that make it

suitable for most XML processing requirements. Some of these options are only available in the XMLNSC parser. Although the XMLNSC parser can parse XML documents without an XML schema, extra features of the parser become available when it operates in model-driven mode. In model-driven mode, the XMLNSC parser is guided by an XML schema, which defines the structure of the message tree (also known as the logical model). The logical message model is equivalent to the WebSphere Enterprise Service Bus Service Message Object structure.

- The DFDL parser reads and writes a wide variety of message formats, and is intended for general text and binary message formats, including industry standards. It is not intended for parsing and writing XML or JSON formatted messages, which have their own message domains. When reading a message, the DFDL parser interprets a bit stream by using grammar that is defined in a DFDL schema file, and generates a corresponding DFDL domain logical message tree in the broker. When writing a message, the DFDL serializer generates a DFDL formatted bit stream from a DFDL domain logical message tree. Because the DFDL parser is model-driven, it can validate messages against the message model that is defined in the DFDL schema file. The level of validation that is performed by the DFDL parser is the same as the level that is defined by XML schema 1.0. For more information, see DFDL parser and domain.
- The DataObject parser reads and writes messages from Enterprise Information Systems (EIS) such as SAP, PeopleSoft, and Siebel. Such messages belong to the DataObject domain. When it receives a message from an adapter, the DataObject parser constructs a message tree from the business object that it receives from the EIS. When it writes a message, the DataObject parser creates from the message tree the business object that it sends to the EIS. The DataObject parser is always model-driven, and it is guided by the XML schemas that model the EIS business objects. The XML schemas are created automatically from the content of a message set. You use the DataObject domain to parse and write messages for WebSphere Adapters and CORBA applications.

Each parser is suited to a particular class of message format that is called a domain. For more information, see Parsers.

IBM Integration Bus parsers are analogous to the WebSphere Enterprise Service Bus data handlers, in that they are responsible for the conversion between the physical message format and a logical message structure. In WebSphere Enterprise Service Bus, the data handler is responsible for populating part of the Service Message Object (SMO) structure, while in IBM Integration Bus, the parser is responsible for populating the message tree.

In IBM Integration Bus, some message formats are self-defining and can be parsed without reference to a model. However, most message formats are not self-defining, and a parser must have access to a message model that describes the message, if it is to parse the message correctly. Message models are based on W3C XML schema. When the message format is known, IBM Integration Bus can parse an incoming message bit stream and convert it into a logical message tree for manipulation by a message flow. After the message has been processed by the message flow, IBM Integration Bus converts the message tree back into a message bit stream. Conversion to and from a bit stream is performed by a parser.

You can model a wide variety of message formats by using XML schema files, DFDL schema files, and WebSphere Adapter schema files. IBM Integration Bus provides importers to read metadata such as C header files, COBOL copybooks, and EIS metadata, and to create message models from that metadata. By using these importers, you speed up the creation of message models. Additionally, predefined models are available for common industry standard message formats such as SWIFT, EDIFACT, X12, FIX, HL7, and TLOG. For more information, see Modeling different data formats.

When you create a message model, you obtain the following benefits:

- Runtime validation of the messages: A parser checks whether the input and the output messages have the correct structure and data values.
- Enhanced parsing of the XML messages: The parser is provided with the data type of data values, and can cast the data accordingly.
- Improved productivity when writing ESQL: The ESQL editor can use message models to provide code completion assistance.
- Drag-and-drop operations on message maps: When you are creating message maps, the Graphical Data Mapping editor uses the message model to populate its source and target views.
- Reuse of message models, in whole or in part: You can create additional messages that are based on existing messages.
- Generation of documentation.
- Provision of version control and access control for message models by storing them in a central repository.

For more information, see Constructing message models.

Data Format Description Language (DFDL)

Data Format Description Language (DFDL) is a modeling language from the Open Grid Forum that is based on XML schema 1.0. DFDL uses standard XSD model objects to express the logical structure of the data, together with DFDL annotations to describe the text or binary physical representation. You use DFDL to describe many data formats:

- Textual and binary
- Commercial record-oriented
- Scientific and numeric
- Modern and legacy
- Industry standards

You use the DFDL schema editor to create, edit, and test DFDL message models within the IBM Integration Toolkit. For more information, see DFDL schema editor.

IBM Integration Bus provides support for a DFDL domain. The DFDL domain can be used to parse and write a wide variety of message formats, and is intended for general text and binary message formats, including industry standards. It is not intended for parsing and writing XML or JSON formatted messages, which have their own message domains. For more information, see DFDL parser and domain.

Patterns framework

IBM Integration Bus provides more built-in patterns that speed up the process of implementing your integration solutions. In addition, you can create your own

user-defined patterns to extend the functionality of IBM Integration Bus. IBM Integration Bus provides the capabilities to graphically define and contribute your own patterns to the Patterns Explorer.

Prebuilt patterns

A pattern is a reusable solution that encapsulates a top-down tested approach to solving a common architecture, design, or deployment task in a particular context. Patterns provide the following benefits:

- Give you guidance for the implementation of solutions.
- Increase development efficienc. Resources are generated from a set of predefined templates.
- Result in higher quality solutions, through reuse of assets and common implementation of programming approaches, such as error handling and logging.

A number of patterns are supplied in the IBM Integration Toolkit, and you can use the Patterns Explorer, which includes comprehensive help, to simplify creation of common scenarios. For more information, see Built-in patterns. You can configure these patterns with values for use in your own environment to solve specific business problems. The supplied patterns use preferred techniques in message flow design to produce efficient, and reliable flows. For more information, see Patterns.

You can create user-defined patterns to extend the functionality of IBM Integration Bus. For more information, see Creating a user-defined pattern.

User-defined patterns

User-defined patterns extend the function of IBM Integration Bus so that you can create patterns that you can reuse within your organization. For more information, see User-defined patterns.

You can package your user-defined pattern into a pattern archive so that the user-defined pattern can be distributed to pattern users by adding the pattern archive to a pattern community site. For more information, see Packaging and distributing pattern plug-ins.

Operational management

IBM Integration Bus provides the following workload management capabilities:

- Allows system administrators to monitor and adjust the speed at which messages are processed.
- Allows system administrators to control the actions to take on unresponsive flows and threads.
- Allows system administrators to specify a workload management policy.

Message flow notification

The system administrator can set the notification threshold for individual message flows deployed to manage their workload. An out of range notification message is produced if the notification threshold is exceeded. A back in range notification message is produced if the notification threshold later drops back into range. For more information, see Configure a message flow to cause notification threshold messages to be published.

Message flow maximum rate

The system administrator can set the maximum rate that an individual message flow can run at. The maximum rate is specified as the total

number of input messages processed every second. When set, the number of input messages that are processed across the flow is measured. This measure is irrespective of the number of additional instances that are used, the number of input nodes in the message flow, or the number of errors that occur. If necessary, a processing delay is introduced to keep the input message processing rate under the maximum flow rate setting. For more information, see Setting the maximum rate for a message flow.

Unresponsive message flows

The system administrator can specify and monitor the maximum amount of time that any message flow is allowed to process a message for, and can specify an action to be taken if the timeout is exceeded. Additionally, manual requests can be made to stop a message flow by restarting the execution group. For more information, see Unresponsive message flows.

Configure a policy

In IBM Integration Bus, a *configurable service* can be used to configure a set of attributes that can be accessed at run time. A system administrator can change the values of the attributes for a configurable service, which then affects the behavior of an IBM Integration Bus node or message flow without the need for redeployment.

The system administrator can specify a workload management policy under a configurable service for a message flow. The workload management policy has the advantage in that it encompasses all of the properties available under workload management in one place and therefore allows for easier tuning of message flow performance. For more information, see Configure a workload management policy within Integration Registry.

.NET

In IBM Integration Bus, you can use the Microsoft .NET Framework (.NET) support to host and run .NET applications and code.

- To interact with other applications that have .NET or Component Object Model (COM) interfaces and perform tasks such as message enrichment, by obtaining data from these applications, you can use the .NETCompute.
- You can call .NET methods directly from ESQL. For more information, see CREATE FUNCTION statement and CREATE PROCEDURE statement.
- To enable Microsoft applications to be rapidly integrated with IBM Integration Bus integration solutions, you can use prebuilt patterns for Microsoft Dynamics CRM that allow transformation and synchronization of client data with another CRM application.
- To receive data from other applications that have Microsoft .NET or Component Object Model (COM) interfaces, you can use a .NETInput node.

.NETCompute node

You can use the .NETCompute node to route or transform messages by using any Common Language Runtime (CLR) compliant .NET programming language, such as C#, Visual Basic (VB), F# and C++/CLI (Common Language Infrastructure). By using this node, IBM Integration Bus can interact with other applications that have .NET or Component Object Model (COM) interfaces and perform tasks such as message enrichment, by obtaining data from these applications. For more information, see .NETCompute node and Using .NET.

.NET prebuilt patterns

IBM Integration Bus provides prebuilt patterns for Microsoft Dynamics CRM that allow transformation and synchronization of client data with another CRM application. These patterns enable Microsoft applications to be rapidly integrated with IBM Integration Bus integration solutions.

- A prebuilt pattern assists with the integration of Microsoft Dynamics CRM with a SAP system. Updates to customer data from create, read, update, and delete operations using a SAP BAPI can be easily reflected in a matching Dynamics CRM Account Entity. For more information, see Microsoft Dynamics CRM Account Entity output: Static BAPI input.
- A prebuilt pattern builds upon the first by allowing a Microsoft Dynamics CRM application to be updated by using various other mechanisms, including WebSphere MQ messages, HTTP, or flat files. For more information, see Microsoft Dynamics CRM Account Entity output: Dynamic transport input.

.NETInput

The .NETInput node provides an easy way to drive message flows that can receive data from other applications that have Microsoft .NET or Component Object Model (COM) interfaces. The .NETInput node provides C#, Visual Basic, and Visual F# templates for Microsoft Visual Studio, which when implemented, connect and retrieve the data for the node. For example, the .NETInput node can be used to retrieve messages from MSMQ. For more information, see .NETInput node and Using .NET.

You can also create a template from an instance of a .NETInput node in a message flow. The result is a Cloned node that has its own icon, properties and runtime implementation, which can be easily exported and shared with other IBM Integration Toolkit developers. For more information, see Cloning a .NETInput node.

File processing

In IBM Integration Bus, you use message flow nodes to handle data from files. You use a FileInput node to process messages that need to read data from files, a FileOutput node to write messages to files, and a FileRead node to read one record, or the entire contents of a file, from within a message flow.

For more information, see How the broker processes files.

WebSphere ESB conversion tool

IBM Integration Bus supplies a WebSphere ESB conversion tool that accelerates the conversion of development artefacts created for WebSphere Enterprise Service Bus in WebSphere Integration Developer or IBM Integration Designer to IBM Integration Bus development artefacts.

For more information, see The WebSphere ESB conversion tool and Converting WebSphere Enterprise Service Bus resources by using the WebSphere ESB conversion tool.

Application development differences

When you develop integration solutions in IBM Integration Bus, you must consider the following differences:

WSDL files

In IBM Integration Bus, each inline schema embedded in your WSDL file is externalized. You must create a schema (.xsd) file for each inline schema.

The WebSphere ESB conversion tool converts your WebSphere Enterprise Service Bus WSDL file and creates a schema (.xsd) file for each inline schema embedded in your WSDL file. The format of the converted WSDL file is different to the original, but it is logically equivalent.

To learn more about WSDL files in IBM Integration Bus, see Working with WSDL.

Integration solution

You convert a WebSphere Enterprise Service Bus mediation module into an IBM Integration Bus application or integration service.

For more information, see Choosing the target integration solution type.

Exports and imports

In IBM Integration Bus, you use a message flow node to recreate an export or an import. The node is specific to a type of transport. The transport binding is configured automatically and determined by the type of node that you use.

To specify the parser that you want to use to serialize or deserialize your data in an export or import, you must configure the **Message Domain** property.

To handle faults, you connect the Fault terminal of a node and implement its logic.

For exports that use a SOAPInput node, the function selector logic is provided by the node.

For all other exports, you must manually implement the function selector as routing logic within the message flow. This can be achieved by using the IBM Integration Bus capabilities for querying the message content and selecting the route within the flow, for example by using a JavaCompute node and a RouteToLabel node.

For more information on IBM Integration Bus built-in nodes, see Built-in nodes.

For more information, see Converting WebSphere Enterprise Service Bus resources manually.

Invocation styles to other services

In IBM Integration Bus, you define the invocation style of a call to a service provider by using a Request message flow node within a message flow. Request-response operations can be called synchronously or asynchronously. IBM Integration Bus provides a synchronous and an asynchronous implementation for Request and Response message flow nodes.

For more information, see Converting SOAP/HTTP WebSphere Enterprise Service Bus service invocation styles manually and Converting SOAP/JMS WebSphere Enterprise Service Bus service invocation styles manually.

Graphical data maps

WebSphere Enterprise Service Bus and IBM Integration Bus provide built-in transform types that you can use to construct your transformation maps. WebSphere Enterprise Service Bus group, lookup, relationship, and relationship lookup transforms do not have equivalent transform types in IBM Integration Bus. You must recreate these transforms manually.

For more information, see Mapping transform equivalents between the Graphical Data Mapping editor and the WebSphere Enterprise Service Bus Graphical Mapping editor.

Handling data from files

In WebSphere Enterprise Service Bus, you use the WebSphere Adapter for Flat Files to exchange data with the local file system. In IBM Integration Bus, you use message flow nodes to handle data from files. For this reason, capabilities to handle data from files, that you have in WebSphere Enterprise Service Bus on the WebSphere Adapter for Flat Files, are located in IBM Integration Bus on the message flow nodes.

IBM Integration Bus provides three different primitive nodes for handling data from files:

- FileInput node: Use the FileInput node to process messages that are read from files. For more information, see FileInput node.
- FileOutput node: Use the FileOutput node to write messages to files. For more information, see FileOutput node.
- FileRead node: Use the FileRead node to read one record, or the entire contents of a file, from within a message flow. For more information, see FileRead node.

Each file message flow node provides four Record detection options which describe how data should be read from a file, written to a file. and propagated through the message flow. Based on the option that you choose, you decide how much of the file content is carried through the message flow as part of a single propagation, either the whole file, a fixed length of data, a record until a delimiter is found, or a section of data which conforms to a predefined message model (parsed record sequence).

In IBM Integration Bus, the combination of a built-in parser, for example BLOB, XMLNSC, DFDL, JSON, and a Record detection option is the equivalent to the WebSphere Adapter for Flat Files built-in data handlers in WebSphere Enterprise Service Bus.

IBM Integration Bus file nodes also provide support for commands to run before or after an FTP or SFTP transfer finishes. For more information, see Local environment overrides for the remote server on the FileOutput node
.

User-defined patterns

WebSphere Enterprise Service Bus and IBM Integration Bus provide built-in patterns to help you solve a specific business problem and the capability to create your own user-defined patterns. In WebSphere Enterprise Service Bus, the process is more complex and requires more coding. In IBM Integration Bus , you can define graphically your user-defined pattern and add it to the Patterns Explorer.

Terminology differences

To learn about the terminology differences between IBM Integration Bus and WebSphere Enterprise Service Bus, see WebSphere Enterprise Service Bus conversion.

DataObject

The term *DataObject* is used for different purposes in WebSphere Enterprise Service Bus and IBM Integration Bus. In WebSphere Enterprise Service Bus, DataObject refers to a part of the Service Data Object (SDO) API `commonj.sdo.DataObject`, which is used to represent structured data elements within the Service Message Object. In IBM Integration Bus, the term DataObject refers to the DataObject parser, which reads and writes messages from Enterprise Information Systems (EIS) such as SAP, PeopleSoft, and Siebel.

Runtime environment differences

In IBM Integration Bus, you will find the following runtime environment differences:

- IBM Integration Bus requires WebSphere MQ to run successfully.

  For more information, see Additional software requirements.
- IBM Integration Bus does not require WebSphere Application Server, nor does it run inside an Application Server container. It runs as a standalone set of operating system processes.

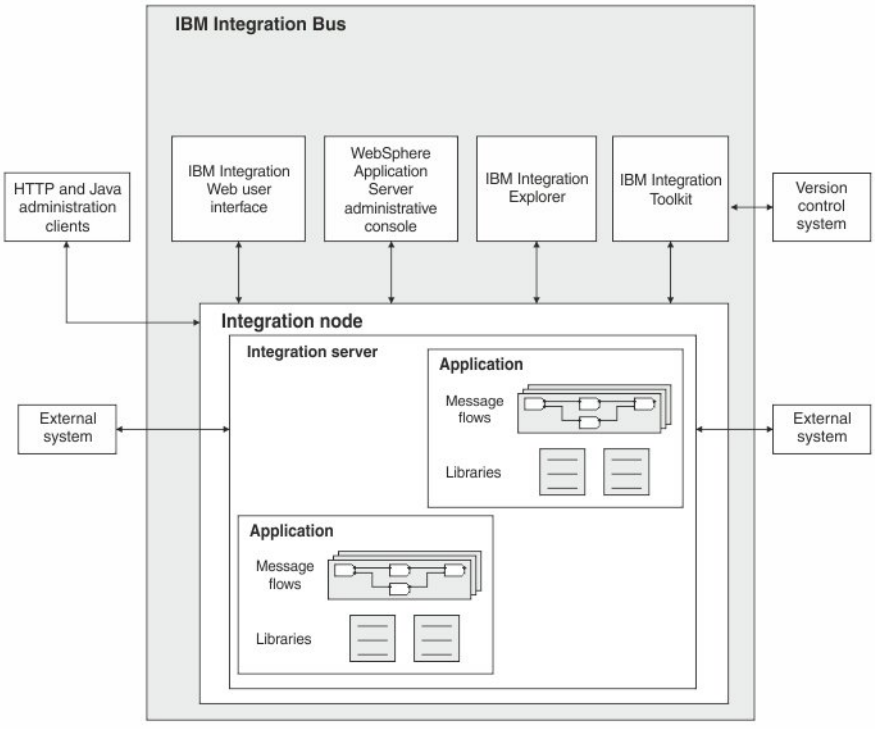
Chapter 4. Name changes in IBM Integration Bus Version 9.0

IBM Integration Bus is a compatible evolution of WebSphere Message Broker that has also been designed to incorporate the features found in WebSphere Enterprise Service Bus.

IBM Integration Bus provides a universal integration capability which addresses a wide range of integration scenarios including web services (SOAP and REST), messaging, database, file, ERP systems, mobile, physical devices, email, custom systems and more.

An integration bus is a collection of *integration nodes*. An integration node can be a broker, for example, or a WebSphere Application Server node. This concept is reflected in the names of various resources; for example, the name of the default broker in IBM Integration Bus Version 9.0 is IB9NODE.

The following diagram illustrates the architecture of IBM Integration Bus Version 9.0.



The following table lists the resource names that have changed in IBM Integration Bus Version 9.0.

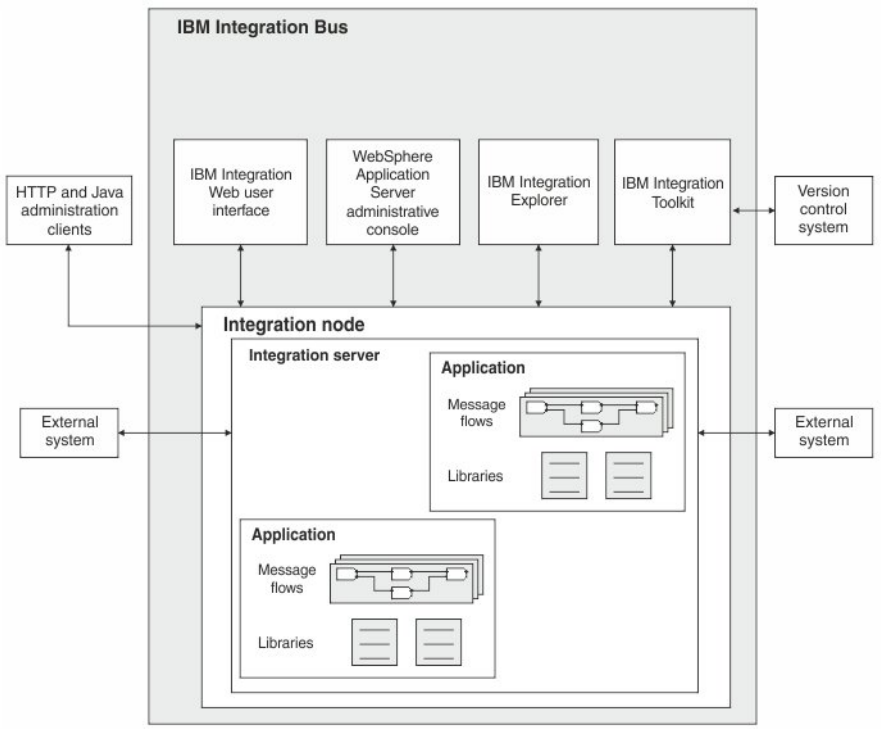
IBM Integration Bus terms	WebSphere Message Broker terms
IBM Integration Toolkit	WebSphere Message Broker Toolkit
IBM Integration Explorer	WebSphere Message Broker Explorer
IBM Integration Administration for WebSphere Application Server	WebSphere Message Broker Administration for WebSphere Application Server

IBM Integration Bus terms	WebSphere Message Broker terms
IBM Integration web user interface	WebSphere Message Broker web user interface
Integration node	Broker
Integration broker (shortened to broker)	Broker
Integration server	Execution group
Integration service	Service
Integration Bus component	Broker component
Integration API	Message Broker API
Integration Java API	Message Broker Java API
IB9NODE (default broker name)	MB8BROKER (default broker name)
IB9QMGR (default queue manager name)	MB8QMGR (default queue manager name)
Integration Development perspective	Broker Application Development perspective
Application Development view	Broker Development view
Integration Nodes view	Brokers view
Integration project	Message Broker project

For a summary of the features that are new to IBM Integration Bus Version 9.0, see Chapter 3, “What's new in Version 9.0?,” on page 5.

Chapter 5. IBM Integration Bus technical overview

IBM Integration Bus enables information packaged as messages to flow between different business applications, ranging from large traditional systems through to unmanned devices such as sensors on pipelines.



IBM Integration Bus processes messages in two ways: message routing and message transformation.

Message routing

Messages can be routed from sender to recipient based on the content of the message.

The message flows that you design control message routing. A message flow describes the operations to be performed on the incoming message, and the sequence in which they are carried out.

Each message flow consists of the following parts:

- A series of steps used to process a message; see Message flow nodes.
- Connections between the nodes, defining routes through the processing; see Message flow connections.

IBM supplies built-in nodes and samples for many common functions. If you require additional functions, you can write your own user-defined nodes; see User-defined extensions overview.

You create message flows in the IBM Integration Toolkit.

Message transformation

Messages can be transformed before being delivered:

- They can be transformed from one format to another, perhaps to accommodate the different requirements of the sender and the recipient.
- They can be transformed by modifying, combining, adding, or removing data fields, perhaps involving the use of information stored in a database. Information can be mapped between messages and databases. More complex manipulation of message data can be achieved by writing code, for example in Extended SQL (ESQL) or Java, within configurable nodes.

Transformations can be made by various nodes in a message flow. Before a message flow node can operate on an incoming message, it must understand the structure of that message.

- Some messages contain a definition of their own structure and format. These messages are known as self-defining messages, which you can handle without the need for additional information about structure and format; see Self-defining elements and messages.
- Other messages do not contain information about their structure and format. To process them, you must create a model of their structure; see The message model.

Like message flows, you create message models in the IBM Integration Toolkit. They can contain two types of information:

- The logical structure: the abstract arrangement and characteristics of the data, represented as a tree structure; see The message model.
- One or more physical formats: the way the data is represented and delimited in the physical bit stream; see Message Sets: Physical formats in the MRM domain.

Create the broker environment

The work of routing and transforming messages takes place in a broker. Within the broker, you can define one or more execution groups, which are processes in which message flows run.

The mode in which your broker is working can affect the number of execution groups and message flows that you can deploy, and the types of node that you can use. See Restrictions that apply in each operation mode.

You can install and create one or more brokers on one or more computers that are running a supported operating system. If you create multiple brokers, you can configure your environment to provide protection against failure, and you can separate work across different divisions in a business.

You administer the broker by using product commands, the IBM Integration Explorer, the IBM Integration API within the Integration Nodes view, or the IBM Integration API (also known as the Integration API) in your own applications.

Develop applications

After your system administrator has created your brokers, your application developers can create and modify message flows and message definitions by using the IBM Integration Toolkit.

Different perspectives in the IBM Integration Toolkit are used to develop message flows, message model schema files, and other related resources; see “IBM Integration Toolkit” on page 58.

You can use a repository to provide access control and version control of your development resources. A repository also allows multiple developers to work on the same resources in parallel; see “Development repository” on page 65.

Your applications can communicate with the broker by using a range of protocols that includes WebSphere MQ, JMS 1.1, HTTP and HTTPS, Web Services (SOAP and REST), File, Enterprise Information Systems (including SAP and Siebel), and TCP/IP. For more information about connecting applications, see Nodes for connectivity.

Deploy applications to the runtime environment

When you have created and configured your message flows, message model schema files, and associated resources by using the Integration Development perspective of the IBM Integration Toolkit, you can deploy the executable data to one or more brokers; see Packaging and deployment overview.

You can deploy data in the following ways:

- From the Integration Nodes view of the IBM Integration Toolkit
- From the stand-alone administrative interface, the IBM Integration Explorer
- From the Test Client environment in the IBM Integration Toolkit
- By using a command
- By creating applications that use the IBM Integration API application programming interface

When you deploy message flows and message model schema filess, they are compiled and enveloped in a broker archive (BAR) file, and sent to the target broker; see Packaging and deployment overview. The BAR file has configurable system properties. You can override properties such as queue and database names, without the need to change source files or redevelop the message flow. This configuration makes it easier to move definitions between systems.

The broker opens the BAR file, removes the contents, makes a record of the information that it has received, and discards the envelope. It retains the information in its local storage area within the computer file system, so that it can restore the application resources and restart messages flows if and when required.

Publish/Subscribe

Publish/subscribe is a style of messaging for which IBM Integration Bus provides limited support; in WebSphere Message Broker Version 7.0 this support was transferred to WebSphere MQ. If you have been connecting publish/subscribe applications to brokers in previous versions, see Migrating publish/subscribe information to WebSphere MQ.

Further information

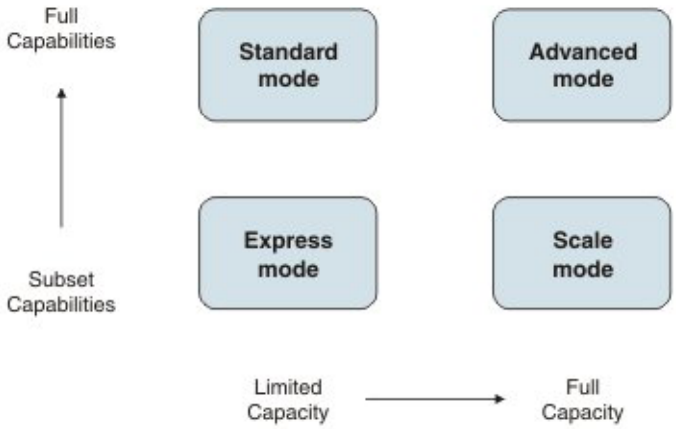
For a basic introduction to IBM Integration Bus, see the IBM Redbooks® publication IBM Integration Bus Basics.

IBM Integration features

Some features of IBM Integration Bus have specific requirements, such as a specific mode of operation or the availability of an additional product, such as a database. Additional terms and conditions also apply to the use of some IBM Integration Bus features, including the requirement for other product licenses.

You can choose the mode of operation for your broker based on the functionality and capacity that you require:

- In `express` mode, the broker operates with a limited set of nodes for use with a single execution group.
- In `scale` mode, the broker operates with a limited set of nodes for use with unlimited execution groups.
- In `standard` mode, the broker operates with all features enabled but you are limited to creating only one execution group.
- In `advanced` mode, the broker operates with all features enabled, and there is no limit to the number of execution groups or the number of message flows deployed to each execution group.



Other modes of operation are also available for use under specific conditions (`trial` mode, `developer` mode, and `adapter` mode); for more information, see "Operation modes" on page 66.

The following table provides an overview of the main IBM Integration Bus capabilities, and shows the modes of operation in which they are available. The table also indicates where additional requirements or restrictions apply.

Table 1. Summary of IBM Integration Bus features

Capability	Description	Express mode	Standard mode	Scale mode	Advanced mode
Vertical scaling	Unlimited execution groups can be used.	No	No	Yes	Yes

Table 1. Summary of IBM Integration Bus features (continued)

Capability	Description	Express mode	Standard mode	Scale mode	Advanced mode
Web Standards connectivity	The following nodes are provided: • HTTPAsyncRequest node • HTTPAsyncResponse node • HTTPHeader node • HTTPInput node • HTTPReply node • HTTPRequest node • SOAPAsyncRequest node • SOAPAsyncResponse node • SOAPEnvelope node • SOAPExtract node • SOAPInput node • SOAPReply node • SOAPRequest node Support is also provided for REST, XML, and JSON.	Yes	Yes	Yes	Yes
Java connectivity and JDBC	The following nodes are provided: • Database node • DatabaseRetrieve node • DatabaseRoute node • JavaCompute node • JMSHeader node • JMSInput node • JMSMQTransform node • JMSOutput node • JMSReceive node • JMSReply node • MQJMSTransform node	Yes	Yes	Yes	Yes
MQ connectivity	The following nodes are provided: • MQGet node • MQHeader node • MQInput node • MQOutput node • MQReply node • Publication node Support is also provided for content-based filtering.	Yes	Yes	Yes	Yes
Graphical mapping and XSL	The following nodes are provided: • Mapping node • XSLTransform node	Yes	Yes	Yes	Yes

Table 1. Summary of IBM Integration Bus features (continued)

Capability	Description	Express mode	Standard mode	Scale mode	Advanced mode
Flow control	The following nodes are provided: • Input node • Route node • RouteToLabel node • Output node • Throw node • TimeoutControl node • TimeoutNotification node • Trace node • TryCatch node • FlowOrder node • Passthrough node • ResetContentDescriptor node • Validate node • Check node • Label node Support is also provided for subflows.	Yes	Yes	Yes	Yes
Email, File, and (S)FTP connectivity	The following nodes are provided: • EmailInput node • EmailOutput node • FileInput node • FileOutput node	Yes	Yes	Yes	Yes
User-defined logic	Support is provided for user-defined nodes, parsers, and exits.	Yes	Yes	Yes	Yes
BPM support	The following nodes are provided: • SCAAsyncRequest node • SCAAsyncResponse node • SCAInput node • SCAReply node • SCARequest node	No	Yes	Yes	Yes
WSRR support	The following nodes are provided: • EndpointLookup node • RegistryLookup node	No	Yes	Yes	Yes
Microsoft .NET	A .NETCompute node is provided.	Yes	Yes	Yes	Yes
Patterns technology	Support is provided for built-in and user-defined patterns technology.	Yes[1]	Yes[1]	Yes[1]	Yes[1]
Administration interfaces	The following administration interfaces are provided: • Web user interface • WebSphere Application Server (WAS) console • Message Broker Explorer • Command line • Message Broker API (CMP) • Representational State Transfer (REST) API	Yes	Yes	Yes	Yes
Record and replay	Support is provided for recording and replaying messages.	Yes[2]	Yes[2]	Yes[2]	Yes[2]
Built-in caching	A Broker-embedded cache uses WebSphere Extreme Scale technology.	Yes	Yes	Yes	Yes

Table 1. Summary of IBM Integration Bus features (continued)

Capability	Description	Express mode	Standard mode	Scale mode	Advanced mode
Mobile integration	Mobile integration patterns are provided.	Yes[1, 3]	Yes[3]	Yes[1, 3]	Yes[3]
High availability	Support is provided for multi-instance brokers.	Yes[4]	Yes[4]	Yes[4]	Yes[4]
PHP	A PHPCompute node is provided.	Yes	Yes	No	Yes
TCP/IP	The following nodes are provided: • TCPIPClientInput node • TCPIPClientOutput node • TCPIPClientReceive node • TCPIPServerInput node • TCPIPServerOutput node • TCPIPServerReceive node	Yes	Yes	No	Yes
SQL	The following nodes are provided: • Compute node • DatabaseInput node Support is also provided for ODBC database and ESQL deployment.	No	Yes	No	Yes
Advanced data processing	The following nodes are provided: • AggregateControl node • AggregateReply node • AggregateRequest node • Collector node • Filter node • Resequence node • Sequence node	No	Yes	No	Yes
Advanced Policy Enforcement Point (PEP) processing	A SecurityPEP node is provided.	No	Yes	No	Yes
Built-in application adapters	The following nodes are provided: • JDEdwardsInput node • JDEdwardsRequest node • PeopleSoftInput node • PeopleSoftRequest node • SAPInput node • SAPReply node • SAPRequest node • SiebelInput node • SiebelRequest node • TwineballInput node • TwineballRequest node	No	Yes	Yes[5]	Yes
Managed file support	The following nodes are provided: • CDInput node • CDOutput node • FileRead node • FTEInput node • FTEOutput node	No	Yes	No	Yes
CICS, IMS, CORBA	The following nodes are provided: • CICSRequest node • CORBARequest node • IMSRequest node	No	Yes	No	Yes

Notes:

1. Assets created using the patterns technology can use only the features to which you are entitled under the terms of your license.
2. Record and replay capabilities require a supported database.
3. Mobile integration is optimized for use with Worklight Server.
4. Separate licenses are required for Warm Standby instances.
5. A separate license for adapters is required before you can use the WebSphere Adapters nodes in Scale mode.

IBM Integration Toolkit

The IBM Integration Toolkit is an integrated development environment and graphical user interface based on the Eclipse platform.

Application developers work in separate instances of the IBM Integration Toolkit to develop resources associated with message flows. The IBM Integration Toolkit connects to one or more brokers to which the message flows are deployed.

You can install the IBM Integration Toolkit only on Windows and Linux on x86. You can only view and interact with brokers that you have created in IBM Integration Bus Version 9.0.

The IBM Integration Toolkit

When you start the IBM Integration Toolkit, a single window is displayed. This window is the IBM Integration Toolkit, which contains one or more perspectives.

A perspective is a collection of views and editors that you use to complete a specific task, or work with specific types of resource. The two significant perspectives in the IBM Integration Toolkit are the Integration Development perspective for application development, and the Debug perspective for debugging message flows. The first time that you start the IBM Integration Toolkit, the Integration Development perspective is displayed.

An additional stand-alone component, the IBM Integration Explorer, is supplied for advanced administrative users, and enables additional administration tasks that you cannot perform in the IBM Integration Toolkit.

Accessing context-sensitive help

The Help view provides context-sensitive help throughout the IBM Integration Toolkit. You can display the Help view for most aspects of the user interface (for example, on the Application Development view, the Message Flow editor, or a properties page) by bringing focus to the object and pressing F1 (on Windows) or SHIFT+F1 (on Linux). The Related Topics page shows description and help topics that are related to the selected object. The About section shows context help that is specific to your current context, and the Dynamic Help section shows some search results that might be related.

Use the other pages in the Help view to view and search the contents of the information center. The All Topics page shows the table of contents of all the books in the information center. The Index provides an index of keywords of all the books in the information center. You can enter a keyword in the text field on the Index page to highlight the best match in the list of keywords. You can use the

Search page to locate topics, samples, and remote documents using keywords in a search query. You can bookmark topics and other documents of interest, and view them in the Bookmarks page.

For a basic introduction to using the IBM Integration Toolkit, see the IBM Redbooks publication IBM Integration Bus Basics.

IBM Integration Toolkit perspectives

A perspective is a group of views and editors that shows various aspects of the resources in the IBM Integration Toolkit.

You can switch perspectives, depending on the task at hand, and customize the layout of views and editors. Switch between perspectives by clicking **Window** > **Open Perspective** > **Other**, then clicking the perspective to which you want to switch.

The IBM Integration Toolkit offers the following perspectives:

Integration Development perspective

The Integration Development perspective is the default perspective that is displayed the first time that you start the IBM Integration Toolkit.

Application developers work in this perspective to develop and modify applications, libraries, message model schema files, message flows, and other associated resources. You can also import relational database schemas for ESQL content assist and validation, and interact with databases by using the Data Project Explorer view and Data Source Explorer view.

The preceding figure shows the Integration Development perspective with a message flow open in the Message Flow editor.

You can use the Integration Nodes view in the Integration Development perspective to create new brokers and deploy resources to connected brokers. Some of the administrative tasks that are available through the IBM Integration Explorer, which is supplied as a separate component that you can install on computers on which you intend to perform only administrative tasks, are also supported by the Integration Nodes view.

Debug perspective

The Debug perspective is where application developers test and debug message flows.

Plug-in Development perspective

The Plug-in Development perspective is where application developers develop plug-ins for user-defined extensions.

Data Analysis

The Data Analysis is where application developers develop models for analyzing complex XML data.

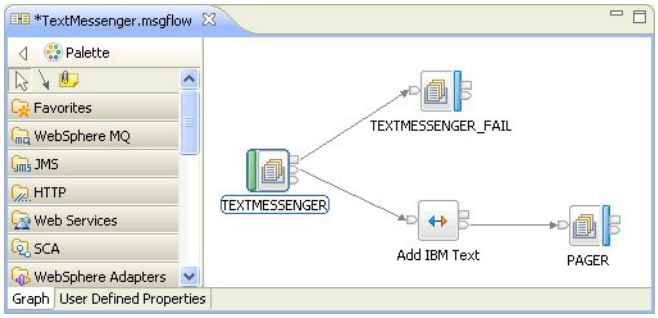
Editors

An editor is a component of the IBM Integration Toolkit. Editors are typically used to edit or browse resources, which are the files, folders, and projects that exist in the workbench.

When you open a file for editing, for example by double-clicking it in the Application Development view, the default editor associated with that file opens in the editor area of the current perspective. By default, the editor area is in the upper-right corner of the IBM Integration Toolkit window.

Open resources with the default editor because other editors might not validate the changes correctly.

The following diagram shows the `TextMessenger.msgflow` file from the Pager sample opened in the Message Flow editor, which is part of the Integration Development perspective.



You can open any number of editors at the same time, but only one editor is active at a time. The main menu bar and main toolbar display the operations that apply to the active editor. By default, editors are stacked in the editor area, but you can tile them to view source files simultaneously. Tabs in the editor area indicate the names of the resources that are open for editing. An asterisk (*) indicates that an editor has unsaved changes. If you attempt to close the editor or exit the IBM Integration Toolkit with unsaved changes, you are prompted to save the changes.

To find out about the Pager sample, click the following link:

- Pager

You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.

Resources

The projects, folders, and files that you work with in the IBM Integration Toolkit workspace are called *resources*. By default, these resources are stored with their metadata in the workspace directory in your local file system. The workspace directory is created the first time that you start the IBM Integration Toolkit.

The default locations for the workspace are in the following places:

- On Linux on x86, the default workspace directory is created at `/home/user_ID/IBM/IntegrationToolkit90/workspace`.
- On Windows Server 2008, the default workspace directory is created at `C:\Users\user_ID\IBM\IntegrationToolkit90\workspace`.

  where *user_ID* is the user name with which you are logged on.

You can create projects in other directories in addition to the workspace directory. You can maintain multiple workspaces by specifying a new location at the prompt when you start your IBM Integration Toolkit session.

Typically, you edit and view IBM Integration Toolkit resources in the Application Development view in the Integration Development perspective.

Resource editors do not automatically reflect the changes that you make in one window in additional windows that you have opened to view the same resource. Close and reopen additional windows each time that you update a resource in an editor session.

Types of resource

You can create and work with three basic types of resource:

Source files

Source files are included in your workspace and are accessible from the Integration Development perspective. When you create source files, they are grouped by file type in *folders*, in the same way as the directories of a file system. You can also group folders or files in *projects*. Projects are used for building, version control, sharing, and resource organization.

The following source files can exist:

- `.msgflow` files contain the message nodes and connections that you create to form a message flow. These files are displayed in the Integration Development perspective in a folder called Flows. The Flows folder is always created and displayed.
- `.subflow` are optional and contain the message nodes and connections that you create to form a message flow that is embedded in a parent message flow. If you have created subflows, they are displayed in the Integration Development perspective in a folder called Subflows.
- `.esql` files are optional and contain ESQL code that performs specific processing steps within a message node. You can code ESQL to tailor message processing in Compute, Database, and Filter nodes. If you have created ESQL files, they are displayed in the Integration Development perspective in a folder called ESQLs.
- `.map` files are optional and map the relationships between different data sources (typically messages and database tables) that you have created by using the graphical mapping tools. You can create maps to tailor message processing in Mapping nodes. If you have created mapping files, they are displayed in the Integration Development perspective in a folder called Maps.
- `.category` files are optional and contain a group of messages, either related to WSDL operations, or to a user-specific purpose or message flow. If you have created category files, they are displayed in the Integration Development perspective in a folder called Message Categories.
- `.mxsd` files are optional and contain definitions of the messages you have modeled. If you have created message models, they are displayed in the Integration Development perspective in a folder called Message Definitions.
- `.wsdl` files are optional and contain WSDL definitions which you have imported into the workspace to use as a source for message definitions.

If you have created WSDL files, they are displayed in the Integration Development perspective in a folder called Deployable WSDL, and are grouped by namespace.

- `.insca` and `.outsca` files are optional Broker SCA definition files. A `.insca` file contains an SCA import component and is used to configure SCAInput and SCAReply nodes. A `.outsca` file contains an SCA export component and is used to configure SCAAsyncRequest, SCAAsyncResponse, and SCARequest nodes. They are displayed in the Integration Development perspective in a folder called Broker SCA Definitions, in the message set project.

You can create resources from a pattern more than once to give unique pattern instances with different configurations, see Patterns. The resources for each pattern instance are contained within a single pattern instance project. The pattern instance project contains links to all projects containing the resources that are created as a result of generating a pattern instance from your configuration, such as message flows, Java classes for JavaCompute nodes, ESQL modules, message maps, test client, XML files, and style sheet files.

Helper files

Helper files maintain information that supports other activities:

- `.broker` files contain definitions of broker connections.
- `.bar` files contain deployable and other files that you have chosen to send to a broker.
- `.mbtest` files contain the steps that define a test that you use with the Test Client to debug your applications.

Deployable files

Deployable files are included in a broker archive (BAR) file and deployed to the broker where they are involved in some way in the processing of messages. A BAR file might contain the following files:

- A `.appzip` file for each application. This file contains all resources that belong to the application, such as `.msgflow`, `.cmf`, `.esql`, `.map`, `.xsd`, and any message set `.dictionary` and `.xsdzip` files. If an application refers to one or more libraries, `.libzip` files for the referenced libraries are also added to the BAR file.
- A `.libzip` file for each library. This file contains all resources that belong to the library, such as `.msgflow`, `.cmf`, `.esql`, `.map`, `.xsd`, and any message set `.dictionary` and `.xsdzip` files. If a library refers to other libraries, `.libzip` files for the referenced libraries are also added to the BAR file.
- A `.cmf` file for each message flow, if in the Broker Archive editor you have selected **Compile and in-line resources**. This file is a compiled version of the message flow. You can have any number of these files within your BAR file.
- A `.msgflow` file for each message flow, unless in the Broker Archive editor you have selected **Compile and in-line resources**. These files contain definitions for message flows and are not compiled.
- One or more `.subflow` files. These files contain definitions for subflows and can be deployed as individual resources. These files are not compiled.
- A `.appdomainzip` file for each AppDomain. These files contain .NET assemblies used by the message flow.

- A `.dictionary` file for each message set dictionary. You can have any number of these files within your BAR file.
- One or more XSD compressed files (`.xsdzip`), if XML schema and WSDL are defined within a message set.
- A `broker.xml` file. This file is called the *broker deployment descriptor*. You can have only one of these files within your BAR file. This file, in XML format, is contained in the `META-INF` folder of the compressed file and can be modified by using a text editor or shell script.
- One or more XML files (`.xml`), style sheets (`.xsl`), and XSLT files (`.xlst`), if required by nodes in the message flows you have added to this BAR file. The XSLTransform node is one that might require these files.
- One or more JAR files, if required by JavaCompute nodes in the message flows you have added to this BAR file.
- One or more inbound or outbound adapter files (`.inadapter` or `.outadapter`), if required by WebSphere Adapter nodes (for example, the SiebelInput node) in the message flows you have added to this BAR file.
- One or more PHP script files (`.php`), if required by PHPCompute nodes in the message flows you have added to this BAR file.
- A `.esql` file for each ESQL file, unless in the Broker Archive editor you have selected **Compile and in-line resources**.
- A `.map` file for each graphical data mapping routine.
- A `.xsd` file for each DFDL and XML schema file in an application or library.
- Other files that you might want to associate with this BAR file. For example, you might want to include Java source files, `.msgflow` files, or `.wsdl` files for future reference. BAR files can contain all files types. If you choose to include source files in the BAR file, source projects for all applications, libraries, and other compiled resources are added to the `src` folder of the BAR file.

Working sets

A *working set* is a logical collection of projects, which you can use to limit the number of resources that are displayed in the Application Development view. By creating and using a working set, you can reduce the visual complexity of what is displayed in the Application Development view, making it easier to manage and work with your projects.

The *active working set* is the current working set of projects that you choose to display.

A working set is created automatically when you create an application or library. When you select a working set, these working sets are enclosed by brackets, for example [Application1]. You can select, create, edit, and delete working sets in the Application Development view. However, you cannot edit the working sets automatically created for applications or libraries. For more details, see Creating a working set.

By name linking

You identify objects using a combination of a *namespace* and a name, referred to as a *fully qualified name*. The use of fully qualified names, called *by name linking*, makes it easy to identify and locate objects, and to correct broken references.

For example, if you rename an object, all references to that object are broken. If you substitute another object with the same name, all the broken references are corrected.

This concept is important when you are working in a team environment. With by name linking, you can share files in a repository and concurrently modify, add, and delete objects in your message flow application. When you integrate the various parts of the message flow application, you can detect and resolve broken references to objects that have been moved, renamed, or deleted.

Project references

When a projects refers to other projects in the workspace, this is called a *project reference*. When one project references another, the files in the referenced project are available for use by the referring project. For example to configure certain nodes, to create libraries of reusable message flows, and to enable Content Assist in the ESQL editor.

The following scenarios in the IBM Integration Toolkit give examples of project references:

- When configuring a Mapping node in a message flow, you must select source and target messages for the map. The messages are contained within one or more message model schema files. In order to select messages, the Integration project must have a reference to the message model schema files. You must also have a reference to a data design project if you want to map to or from a database.
- When configuring a node, such as an MQInput node, you can define the message template for the message type that the node processes. If you have chosen MRM, SOAP, XMLNSC, DataObject, or IDOC as the `Message domain` property of the node, you must also specify the name of the message model schema files that contains the message model. To pre-populate the list of message model schema files in the `Message` property, the Integration project must have a reference to the message model schema files.
- You might want to create a library of reusable ESQL subroutines in an Integration project, or create a library of message flows to reuse in other flows. A message flow that you want to use these subroutines or message flows in, must have a reference to the Integration project from its parent Integration project.
- You can also use project references to enable Content Assist in the ESQL editor. ("Content Assist" is context-sensitive help that displays valid ways in which a code statement can be completed.) If you set up a project reference from an Integration project containing ESQL code to a message model schema files, the ESQL editor is able to display a list of valid message references.
- If you use the Patterns Explorer to generate a pattern instance from a pattern specification, the generated pattern instance project contains references to all the other projects generated from the pattern. See Using patterns.

To create or remove a project reference manually, right-click the project name in the Application Development view and select the **Properties** menu item. Select **Project References** from the Properties window, and a list of all regular projects in the workspace is displayed from which you can select or clear project references. Message flow, message model schema files and pattern instance projects have an additional menu item **Add or Remove Project References** that launches the Add or Remove Project References window where you can select or clear project references.

If you later close or delete a referenced project, or delete an object within it, it is no longer available to the referencing project and an error is generated. You can correct the error by opening the closed project or adding the missing object, with the correct name, and saving.

Development repository

Use a development repository to benefit from features such as version control and access control of files, which make it easier for teams to work on shared resources.

The IBM Integration Toolkit is based on the Eclipse platform, therefore you can access Eclipse-supported repositories directly from the IBM Integration Toolkit. You can use all repositories that are supported by Eclipse with IBM Integration Bus.

You can take the version numbers of resources from the repository, and associate that version number with the message flows and message sets when they are deployed. This association allows you to display which version of the flow is deployed in the IBM Integration Toolkit. An alternative method of assigning a version number to a message flow is to specify one in the IBM Integration Toolkit when the message flow is created; see Message flow version and keywords.

- For information about how to set up the IBM Integration Toolkit to run with CVS, see Configuring CVS to run with the IBM Integration Toolkit.
- For information about how to integrate the Rational® Team Concert client, see Integrating the Rational Team Concert™ client with the IBM Integration Toolkit.
- For information on how to enable Rational ClearCase®, see Configuring the IBM Integration Toolkit to run Rational ClearCase.
- For information about other repositories, read about other team repositories that are supported by Eclipse.

The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Create more than one broker, on one or more computers, to support your applications; creating more than one broker can provide load balancing, or a division of responsibilities. For example, you might have one broker that handles all your financial applications, and another that handles your order processing and fulfillment.

Application programs connect to and send messages to the broker, and receive messages from the broker. Code your applications to use one of the supported protocols for interacting with the broker; for example, WebSphere MQ queues and connections, Web services, or WebSphere Adapters. The broker routes each message by using the rules that you have defined in message flows and message model schema files, and transforms the data into the structure required by the receiving application.

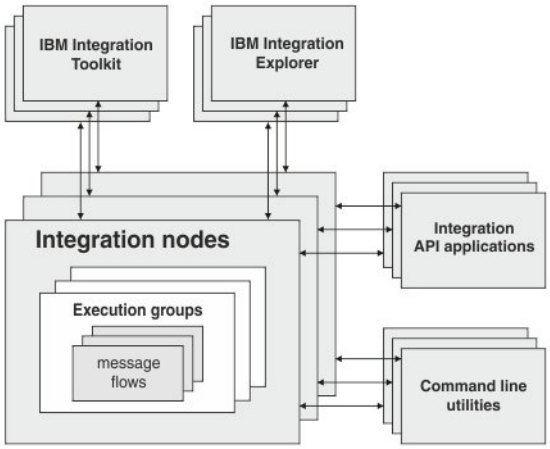
You can install the Integration Bus component on one or more of the supported platforms, which are listed in Operating system requirements. You can create a broker only on the computer on which you have installed the Integration Bus component. You can use the IBM Integration Explorer, the IBM Integration Toolkit, or the command line to create local brokers.

When you create a broker, it operates in one of a number of modes: `advanced`, `adapter`, `standard`, `scale`, or `express`. You must run the broker in the mode that matches the license that you have purchased; see "Operation modes."

Administer the broker by using the IBM Integration Explorer, the Integration Nodes view in the IBM Integration Toolkit, or the product commands. Alternatively, you can write your own programs to use the IBM Integration API (also known as the Integration API).

Manage the application resources of the broker, which include message flows and message model schema files, by using the IBM Integration Toolkit or IBM Integration Explorer; these two applications connect to the broker by using a WebSphere MQ server connection, which is defined to the broker queue manager when you create the broker.

The following figure shows the relationship between the resources that exist at run time, and how they interact with the IBM Integration Explorer and IBM Integration Toolkit.



Resources associated with a broker

When you create a broker, the following resources are also defined and created:

- A WebSphere MQ queue manager, if one does not exist.
- A set of fixed-name queues that are defined to the WebSphere MQ queue manager.
- A default WebSphere MQ SVRCONN channel with a fixed name SYSTEM.BKR.CONFIG, which is used by the IBM Integration Toolkit, the IBM Integration Explorer, and applications that use the IBM Integration API (Integration API).

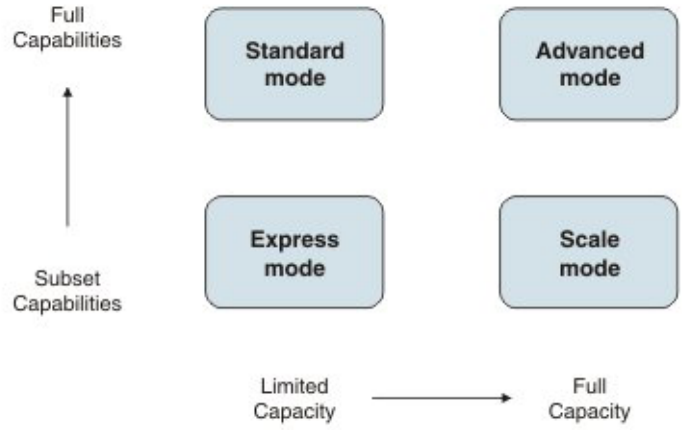
Operation modes

The operation mode that you can use for your broker is determined by the license that you purchase.

The following modes are supported:

- "Express Edition mode" on page 68. A limited set of nodes are enabled for use within a single execution group. Message flows are unlimited.

- "Scale mode" on page 68. A limited set of nodes are enabled for use within unlimited execution groups. Message flows are unlimited.
- "Standard Edition mode" on page 69. All features are enabled for use with a single execution group. The number of message flows that you can deploy are unlimited.
- "Advanced mode" on page 69. All features are enabled and no restrictions or limits are imposed. This mode is the default mode, unless you have the Developer Edition.



- "Developer Edition mode" on page 69. All features are enabled, but you can use the product for evaluation, development, and test purposes only. Developer Edition is available on Windows x86-64 and Linux x86-64 operating systems and is limited to one message (transaction) per second at the message flow level.
- "Trial Edition mode" on page 69. All features are enabled, but you can use the product for only 90 days after installation. Trial Edition is available on all distributed platforms in WebSphere Message Broker Version 8.0.0.0, but has been replaced by the Developer Edition from Version 8.0.0.1 onwards.
- "Remote Adapter Deployment mode" on page 70. Only adapter-related features are enabled, and the types of node that you can use, and the number of execution groups that you can create, are limited.

Operation modes typically restrict the number of nodes that are available, and execution group capacity, but many features are available across all modes of operation. For an overview of the main IBM Integration Bus capabilities, and the modes of operation in which they are available, see "IBM Integration features" on page 54.

In WebSphere Message Broker Version 8.0, Entry Edition was renamed Express® Edition, Starter Edition was renamed Standard Edition, and Enterprise was renamed Advanced.

You must ensure that your brokers are running in the operation mode for which you have purchased a license. You can set the operation mode when you create a broker by using the **mqsicreatebroker** command. On z/OS you can change the operation mode only after you create a broker; see Changing the operation mode of your broker.

If you have purchased a license for the full package, the Standard Edition, or the Remote Adapter Deployment mode, the broker is automatically created in `advanced` mode unless you specify the correct mode for your license.

Change the mode of your broker to conform to your license if necessary; see Changing the operation mode of your broker. You can also report the current mode of your broker; see Checking the operation mode of your broker.

The IBM Integration Toolkit remains the same in all modes. All the capabilities of the IBM Integration Toolkit are available in all modes. If you try to deploy too many message flows or execution groups for the mode, or try to use a node that is not valid in the mode, the operation is rejected, and an error message is displayed indicating the reason for the failure; see Restrictions that apply in each operation mode. Node restrictions for a given mode also apply to the use of message flows generated by IBM Integration Bus patterns.

Fix packs are identical for all modes, and can be applied without affecting the validity of the mode.

Express Edition mode

In `express` mode, the broker operates with a limited set of nodes for use with a single execution group. Message flows are unlimited. For a list of nodes that can be deployed to a broker running in Express Edition mode, see Restrictions that apply in each operation mode.

Because the functions that are enabled and the number of execution groups that you can create are limited, not all samples and patterns work in Express Edition mode. If you want to run samples, see "Development and unit test" on page 70.

Scale mode

In `scale` mode, the broker operates with a limited set of nodes for use with unlimited execution groups. Message flows are unlimited. For a list of nodes that can be deployed to a broker running in Scale mode, see Restrictions that apply in each operation mode.

Because the functions that are enabled are limited, not all samples and patterns work in Scale mode. If you want to run samples, see "Development and unit test" on page 70.

WebSphere Enterprise Service Bus (WESB) customers can obtain a transfer license, which enables you to migrate from WESB to IBM Integration Bus, and to create and run brokers in Scale mode. As a current WESB customer, you can obtain the following transfer licenses:

- IBM Integration Bus WebSphere Enterprise Service Bus Transfer License
- IBM Integration Bus WebSphere Ent Svc Bus Transfer License Idle Standby

These licenses entitle you to obtain and use IBM Integration Bus in Scale mode in exchange for your WESB product licenses.

The license for IBM Integration Bus Standard Edition also entitles you to run brokers in `scale` mode, rather than `standard` mode, if you choose to. (Note: `scale` mode is not supported on z/OS.) For more information about the features that are available in these modes, see "IBM Integration features" on page 54.

Standard Edition mode

In `standard` mode, the broker operates with all features enabled. Use this edition if you expect to use all or most of the features that are available, but intend to configure a limited environment because of low capacity requirements.

You can use all the available functions, but are limited in the number of resources that you can create and maintain. You are limited to creating one execution group; for more information, see Restrictions that apply in each operation mode. If you attempt to exceed the limits of this mode, the deployment is rejected.

You cannot use all the samples when your broker is in `standard` mode, because of the preceding restrictions. If you want to run samples, see "Development and unit test" on page 70.

Advanced mode

In `advanced` mode, the broker operates with all features enabled, and no operational limits on the creation of execution groups or on the number of flows that are deployed to an individual execution group are enforced. If you want to set up a full broker environment that uses most or all the features available, your brokers must operate in this mode, and you therefore require the full license. If you do not specify another mode, your brokers have the mode set to the default value `advanced`.

Trial Edition mode

Trial Edition is available for WebSphere Message Broker Version 8.0.0.0, and has been replaced in Version 8.0.0.1 by the Developer Edition.

In `trial` mode, the broker operates with all features enabled. You can use all available function, and are not limited in the number of resources that you create and maintain. All capability is available for 90 days after installation.

You can download Trial Edition at no charge, from the following website: IBM Integration Bus Trial package.

Developer Edition mode

In `developer` mode, the broker operates with all features enabled. You can use all available function, and you are not limited in the number of resources that you can create and maintain. This edition is available on Windows x86-64 and Linux x86-64 operating systems, and is provided for evaluative purposes only.

Developer Edition is limited to one message (transaction) per second at the message flow level. Each message flow is able to process one message per second, irrespective of the number of input nodes that are attached to the message flow, or the number of additional instances. When using Developer Edition, if a policy is deployed to increase the message rate throughput above one message per second, a message is reported in the syslog / event log stating that the message rate cannot be changed in Developer Edition. The policy that is deployed has no affect on the message rate.

You can download Developer Edition at no charge from the IBM Integration Bus web page, and you are free to use it for as long as you require, within the terms of

the license. Unlike the IBM Integration Bus Trial Edition that was available in previous versions of the product, there is no expiry period for the Developer Edition license.

If you move from WebSphere Message Broker Version 8.0 Trial Edition to IBM Integration Bus Developer Edition, your Trial mode brokers continue to work as before, but they are treated as Developer mode brokers in the Developer Edition, with no expiry. If you subsequently revert from Developer Edition to a Trial Edition, your Trial mode brokers (which were treated as Developer mode brokers in the Developer Edition) revert to Trial mode, complete with the expiry period.

When you have installed Developer Edition, if you subsequently purchase a license and install the full version of IBM Integration Bus, any Developer mode brokers that are started with the full version are treated as Advanced mode brokers. In this case, you must also modify the operation mode of these brokers by using the **mqsimode** command to reflect the license that you have purchased. For more information, see Changing the operation mode of your broker.

Remote Adapter Deployment mode

In `adapter` mode, the broker operates with a limited set of nodes available for deployed flows. Use this edition if you expect your typical use of the broker to be integration with Enterprise Information Systems (EIS). This edition supports the subset of development resources that provide EIS interaction. For a list of nodes that can be deployed to a broker running in Remote Adapter Deployment mode, see Restrictions that apply in each operation mode.

You can create up to two execution groups, with no limit on the number of deployed message flows in each of these execution groups; see Restrictions that apply in each operation mode. If you attempt to exceed the limits of this mode, the deployment is rejected.

You cannot use all the patterns and samples when your broker is in `adapter` mode, because of the preceding restrictions. If you want to run samples, see "Development and unit test."

Development and unit test

Your license also covers use of the product for development and unit test purposes, but check the license to ensure that you conform to any restrictions for development and unit test. You can view the license for IBM Integration Bus by visiting the Software license agreements search website. Search for "IBM Integration Bus" and choose the license that applies to the version you are using.

Contact your IBM representative if you want further details about license agreements, or if you want to purchase additional licenses or change the type of license that you have purchased.

Integration with Tivoli License Manager

If you use IBM Tivoli License Manager to control and manage your licensed software products, you must ensure that you choose the correct license for the IBM Integration Bus edition that you have purchased. For more information, see Installing IBM License Metric Tool.

System management interfaces

The brokers provide a service for independent system management agents.

This service enables a central management facility to access information about a network that includes one or more brokers. Therefore, you can extend your existing system management agents to include IBM Integration Bus resources.

Brokers publish event messages, using fixed topics, in response to configuration changes, state changes, and user actions such as subscription registrations. Brokers also use architected messages to publish events related to their operational status, and changes in that status. These messages are published using the reserved topic root `$SYS` in code page 1208.

An example of a fixed topic is:

```
$SYS/Broker/<brokerName>/Status/ExecutionGroup/<executionGroupName>
```

The topic structure is fixed in this case, but obviously `<brokerName>` and `<executionGroupName>` are replaced with the appropriate values.

An example of the actual message data for this publication is:

```
<Broker uuid="12345678-1234-1234-1234-123456789012">
        <ExecutionGroup uuid="12345678-1234-1234-1234-123456789012">
                  <Stop>
                         <AllMessageFlows/>
                  </Stop>
        </ExecutionGroup>
</Broker>
```

A system management agent can subscribe to these topics, or to a subset of these topics, to receive the detailed information about activity and state changes in the IBM Integration Bus components.

The event messages have a fixed structure, defined in *XML (Extensible Markup Language)*. The format of these messages, constructed in XML, is detailed in The XML message body. The messages cover configuration changes, state changes, error notifications, and detailed subscription and topic information (for example, a subscription registration).

You can develop or purchase system management adapters or customized administrative applications. These applications subscribe to the system management topics generated by IBM Integration Bus to receive information about the activity of its resources.

Execution groups

An execution group is a named grouping of message flows that have been assigned to a broker. The broker enforces a degree of isolation between message flows in distinct execution groups by ensuring that they run in separate address spaces, or as unique processes.

Each execution group is started as a separate operating system process, providing an isolated runtime environment for a set of deployed message flows. Within an execution group, the assigned message flows run in different thread pools. You can specify the size of the thread pool (that is, the number of threads) that are assigned for each message flow by specifying the number of additional instances of each message flow.

The mode that your broker is working in can affect the number of execution groups that you can use; see Restrictions that apply in each operation mode.

A single default execution group is set up ready for use when you create a reference to a broker in the IBM Integration Toolkit. By setting up additional execution groups, you can isolate message flows that handle sensitive data such as payroll records, or security information, or unannounced product information, from other non-sensitive message flows.

If you create additional execution groups, you must give each group a name that is unique within the broker, and assign and deploy one or more message flows to each one.

You can create and deploy execution groups either in the IBM Integration Toolkit, or using commands.

An execution group process is also known as a DataFlowEngine (DFE); this term is typically used in problem determination scenarios (trace contents, diagnostic messages, and so on). A DFE is created as an operating system process, and has a one-to-one relationship with the named execution group. If more than one message flow runs within an execution group, multiple threads are created within the DFE process.

The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

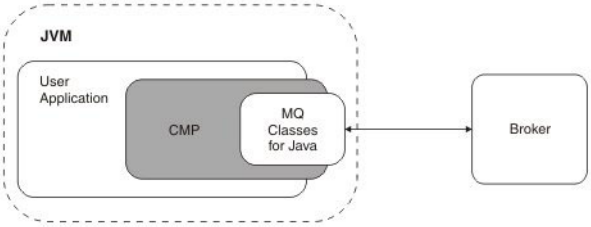
> The IBM Integration API is also known as the Configuration Manager Proxy, or Integration API
>
> The Configuration Manager has been removed from Version 7.0, and the full name of the API has changed. However, the terms CMP application and Integration API have been retained, and are used in this information center to refer to the IBM Integration API, for continuity and consistency with the JAR file `ConfigManagerProxy.jar` that supplies all the classes.

The IBM Integration API (Integration API) consists solely of a Java implementation, and is referred to as the IBM Integration Bus Java API. Your applications have complete access to the broker functions and resources through the set of Java classes that constitute the CMP. Use the Integration API to interact with the broker to perform the following tasks:

- Deploy BAR files
- Change the broker configuration properties
- Create, modify, and delete execution groups
- Inquire and set the status of the broker and its associated resources, and to be informed if status changes
  - Execution groups
  - Deployed message flows
  - Deployed files used by the message flows (for example, JAR files)
- View the Administration log
- View the Activity log
- Create and modify message flow applications. For more information, see Developing message flow applications by using the IBM Integration Java API.

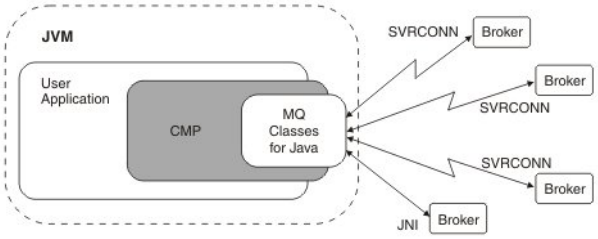
Interaction between CMP applications and the broker

For applications that interact with the broker, the Java classes sit logically between the user application and the broker, inside the Java Virtual Machine (JVM) of the user application. The API requires the WebSphere MQ Classes for Java for connectivity, as shown in the following diagram.



The CMP application can be on the same physical computer as the broker, connected by a JNI (Java Native Interface) connection to the queue manager that uses the WebSphere MQ Java Bindings transport. If appropriate, you can distribute your applications over an Internet Protocol (TCP/IP) network, and connected to the broker by using a WebSphere MQ SVRCONN channel through the WebSphere MQ Java Client transport.

You can use the Integration API to communicate with more than one broker from within the same application, as shown in the following diagram.



Migrating from earlier versions of IBM Integration Bus

For information about migrating your CMP applications to Version 9.0, see Migrating IBM Integration API applications.

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

The IBM Integration Explorer is an extension to the WebSphere MQ Explorer.

You can install the IBM Integration Explorer only on Windows and Linux on x86. You can view and interact only with integration nodes (brokers) that you have created in WebSphere Message Broker Version 7.0 or later.

The IBM Integration Explorer

To use the IBM Integration Explorer, you must start the WebSphere MQ Explorer. The IBM Integration Explorer adds the Integration Nodes folder and Broker Archive Files folder to the MQ Explorer - Navigator view:

- Use the Integration Nodes folder to create, view, and modify integration nodes

- Use the Broker Archive Files folder to import, view, and modify BAR files before deploying them to your integration nodes

If you cannot see these two folders in your MQ Explorer session, you have not installed the plug-ins that are specific to the IBM Integration Bus, which are provided by IBM Integration Explorer. Close your MQ Explorer session, follow the instructions to install the IBM Integration Explorer, then start the MQ Explorer again.

The IBM Integration Explorer provides several QuickViews that you can use to view the properties of integration nodes and their resources. These QuickViews are automatically displayed when you click the resource in the Integration Nodes folder in the MQ Explorer - Navigator view. A QuickView is also available for viewing the details of BAR files that you have imported into the IBM Integration Explorer.

The following views and editors are provided for working with integration nodes in the IBM Integration Explorer:

Broker Archive editor
Use the Broker Archive editor to create and manage broker archive (BAR) files.

Broker Statistics and Broker Statistics Graph views
Use the Broker Statistics and Broker Statistics Graph views to view snapshot accounting and statistics data as it is produced by the broker.

Policy Sets and Policy Set Bindings editor
Use the Policy Sets and Policy Set Bindings editor to edit, save, import, and export policy sets or bindings.

Security Profiles editor
Use the Security Profiles editor to create a security profile for use with Lightweight Directory Access Protocol (LDAP) or Tivoli Federated Identity Manager (TFIM).

DataPower Security wizard
Use the DataPower Security wizard to configure an external DataPower appliance to handle the WS-Security Policy for your HTTP, HTTPS, and SOAP nodes within your message flow.

Administration Log view
Use the Administration Log view to view the results of deployment actions on integration nodes.

Activity Log view
Use the Activity Log view to view recent activities affecting your message flows, and related external resources.

Accessing context-sensitive help

The Help view provides context-sensitive help throughout the IBM Integration Explorer. You can display the Help view for most aspects of the user interface by bringing focus to the object and pressing F1 (on Windows) or SHIFT+F1 (on Linux). The Related Topics page shows description and help topics that are related to the selected object.

The following figure shows the Related Topics page of the Help view that is displayed when you press F1 when the Broker Archive Files folder is selected in the IBM Integration Explorer.

Use the other pages in the Help view to view and search the contents of the information center. The Contents page shows the table of contents of all the books in the information center. The Index provides an index of keywords of all the books in the information center. You can enter a keyword in the text field on the Index page to highlight the best match in the list of keywords. You can use the Search page to locate topics, samples, and remote documents using keywords in a search query. You can bookmark topics and other documents of interest, and view them in the Bookmarks page.

IBM Integration Bus web user interface

The IBM Integration Bus web user interface enables web users to access broker resources through an HTTP client, and provides broker administrators with an alternative to the IBM Integration Explorer for administering broker resources.

You can also use the web user interface to work with statistics and accounting data for your message flows. You can start and stop the collection of snapshot statistics and accounting data, and then display the data in a format that helps you to

analyze and tune the performance of your message flows and applications. For more information, see Viewing accounting and statistics data in the web user interface.

The web user interface also provides support for working with patterns, enabling you to create a new instance of a pattern, configure it, and deploy it.

Support is provided for most major browsers; for more detailed information about supported browsers, see the IBM Support site.

Complete the following steps to configure the web user interface server, create the required web user accounts, and log on to the web user interface:

1. Configure a web user interface server, as described in Configuring the web user interface server.
2. Use the **mqsiwebuseradmin** command to create web user accounts for your web users. For information on how to do this, see Managing web user accounts.
3. Log on to the web user interface server, as described in Accessing the web user interface.
4. Optionally, you can also customize the web user interface according to individual preferences. For more information, see Controlling access to data and resources in the web user interface.

The web user interface is enabled by default for all new brokers when they are created. However, you can disable and enable the web user interface at any time; for more information, see Enabling and disabling the web user interface.

External systems and resources

You can configure IBM Integration Bus resources to interact with a wide range of external systems and resources, such as WebSphere Process Server, and databases.

The products and standards listed here are for guidance only; for details of supported versions, you must check the IBM Integration Bus Requirements Web page.

External systems

Note: From Version 7.5 onwards, WebSphere Process Server has been renamed IBM Business Process Manager Advanced. Information in this topic that refers to WebSphere Process Server Version 7.0 is also applicable to IBM Business Process Manager Advanced Version 7.5. Similarly, from Version 7.5 onwards, WebSphere Integration Developer has been renamed IBM Integration Designer. Information in this topic that refers to WebSphere Integration Developer Version 7 is also applicable to IBM Integration Designer Version 7.5.

IBM Integration Bus interfaces with other IBM products, and with products from other software vendors, to provide services that enhance message processing in the broker:

- IBM WebSphere Service Registry and Repository
- IBM WebSphere Process Server
- IBM WebSphere Integration Developer
- IBM WebSphere Business Monitor
- IBM WebSphere Transformation Extender

- IBM WebSphere MQ File Transfer Edition
- Enterprise Information Systems
  - SAP
  - Siebel
  - Peoplesoft
- Security
  - Tivoli Federated Identity Manager (TFIM)
  - Lightweight Directory Access Protocol (LDAP)
  - RACF® and other External Security Managers, on z/OS only
- Management
  - Tivoli License Manager
- Citrix Presentation Server

External resources

IBM Integration Bus interfaces with other IBM resources, and with resources from other software vendors, to provide extensions to message processing:

- Databases and ODBC support
  - IBM DB2®
  - Oracle
  - Sybase
  - Microsoft SQL Server
  - IBM Informix®
- Databases and JDBC support
  - IBM DB2 Driver for JDBC and SQLJ
  - Microsoft SQL Server 2005 JDBC driver
  - Oracle JDBC Driver
  - Sybase jConnect for JDBC
  - IBM Informix JDBC
- File systems
  - FTP
  - SFTP
- Email systems
- Development repositories
  - Rational ClearCase
  - CVS
  - Other repositories supported by Eclipse
- Compilers (C and Java)
- Browsers
  - Microsoft Internet Explorer
  - Mozilla
- Adobe Acrobat for reading PDF files

External standards

Products and applications that adhere to certain specifications can also interact with IBM Integration Bus:

- Java and JMS providers

- XSD
- WSDL
- SOAP
- XSLT
- WS-Addressing
- WS_Security

For further information about standards, see General industry standards supported by IBM Integration Bus.

Chapter 6. Business scenarios

IBM Integration Bus provides rapid time to value for businesses of all sizes, in all industries and on a range of platforms including cloud and system z.

The top Integration use cases include:

Healthcare integration
Improving efficiency and quality of care within healthcare IT environments, and with the Connectivity Pack for Healthcare integrate clinical applications with bedside medical devices.

Mobile integration
Secure, scalable access to critical data and back-end systems from mobile applications.

.Net integration
Integrate your Microsoft assets with the rest of your services and applications within departmental, small to mid-size or large enterprises.

These topics provide links to a range of business scenarios about using IBM Integration Bus. The scenarios are provided in a variety of formats, including video and articles, hosted on external sites. The main sites for scenarios are:

- IBM Integration YouTube channel
- IBM developerWorks® - WebSphere Message Broker zone

Some scenarios, originally developed for WebSphere Message Broker, apply equally to IBM Integration Bus.

In this information, you can also find topics for samples and patterns that support the variety of scenarios for integration solutions.

- Chapter 7, "Samples," on page 87. The IBM Integration Toolkit provides samples that show the features that are available in IBM Integration Bus, and how to use them.
- Developing integration solutions by using patterns. A *pattern* is a reusable solution that encapsulates a tested approach to solving a common architecture, design, or deployment task in a particular context.

For other information about IBM Integration Bus and its scenarios of use, see the IBM Integration Google+ page.

Healthcare integration scenarios

IBM WebSphere Message Broker Connectivity Pack for Healthcare builds on IBM Integration Bus to provide support for applications in healthcare environments.

In this information, you can find topics for samples and patterns that support the variety of scenarios for healthcare integration solutions.

- Healthcare samples. This sample is a basic example of the Healthcare: HL7 to HL7 pattern, which is included as part of IBM WebSphere Message Broker Connectivity Pack for Healthcare. The pattern mediates between clinical applications that use the HL7 v2 standard for messages. For example, a Patient

Administration System (PAS) might issue messages that are distributed to one or more clinical applications that require the patient information.

- Healthcare patterns. Built-in patterns that are supplied in IBM WebSphere Message Broker Connectivity Pack for Healthcare.

Mobile integration scenarios

IBM Integration Bus, in conjunction with IBM Worklight, provides your mobile applications with secure access to your back-end systems, and integration that is robust and can scale to handle the growing number of connections.

This topic provides links to a range of scenarios about using IBM Integration Bus for mobile integration. The scenarios are provided in a variety of formats, including video and articles hosted on external sites.

Some scenarios, originally developed for WebSphere Message Broker, apply equally to IBM Integration Bus.

Featured scenario: .NET service enablement for mobile applications

Background

Company A is a retail banking business that uses a Microsoft .NET application to work with accounts. The company wants to enable its customers to use mobile devices to perform some actions on their accounts. To do this, the company is looking to reuse the existing .Net code and class, wrapping them as a web service, and providing a mobile application.

The solution

Customers can use a mobile application running on a range of devices to perform actions on their accounts; for example, to query the balance.

By using the IBM Integration Toolkit with the Worklight: Microsoft .NET request-response pattern provided, the application developers create a production-ready solution in a matter of minutes. The solution is based on a web service, which wraps around the existing Microsoft .NET class, made available by an integration application running on IBM Integration Bus. The developers select the methods to be exposed in the web service and, using the pattern and .NET class, the toolkit builds a mobile application and the mobile integration logic written for the Worklight platform.

Implementing the solution can be split into three parts:

Part 1:

1. Create and define the service-enablement logic to be deployed as applications onto mobile devices

a. Create an instance of the Worklight: Microsoft .NET request-response pattern, assigning a unique name to the instance
b. Edit the pattern, to specify parameter values that configure the mobile integration solution; for example, to identify the .Net class.
c. Choose .Net methods that are to be exposed in the service, and configure the parameters for those methods

2. Deploy the integration application, by dragging and dropping onto an execution group. This makes the web service available for applications to call.

This part also generates a mobile application project for use with IBM Worklight, and an Android version of the mobile application for use on mobile devices.

For an illustration of this part, see the video * WebSphere Message Broker - Mobile Service Enablement Part1 (hosted on YouTube)

Part 2:

1. Import into IBM Worklight Studio the project that was generated by the pattern chosen in part 1.
2. Build the mobile application
3. Examine the application artifacts, including:
   - The adapter that allows you to publish or advertise the procedures and functions that mobile applications can call.
   - The procedures to get balance, transfer money, and find missing account
4. Invoke the adapter as a Worklight procedure, to simulate being a mobile application. This sends an HTTP JSON request to Worklight, which passes the request to the adapter, which then packages and sends the request to IBM Integration Bus.
5. Test the mobile application as a mobile web application. By previewing the application as a common resource, you run a working Android application within a web browser.
6. Test the mobile application as an Android application, in the Android emulator provided by Worklight. This runs the application in the same way as it would work on a mobile device.

For an illustration of this part, see the video * WebSphere Message Broker - Mobile Service Enablement Part2 (hosted on YouTube)

Part 3

1. In IBM Worklight Studio, ensure that system parameters and device drivers are installed and set up correctly, primarily the Android SDK and USB Drivers
2. Ensure that your mobile device has the required ADB drivers and set up to deploy and test the mobile application
3. Connect the mobile device to IBM Worklight
4. In IBM Worklight Studio, run the mobile application as an Android application. This installs the application onto the mobile device, and can be used in the same way as in the Android emulator.
5. Disconnect the mobile device, and then use the application as a standalone application on the mobile device

For an illustration of this part, see the video * WebSphere Message Broker - Mobile Service Enablement Part3 (hosted on YouTube)

More scenarios

A selection of more scenarios about using IBM Integration Bus for mobile integration.

WebSphere Message Broker integration with Siebel, SAP, CICS, .Net and Worklight

A two-part video demonstrating a Sales order management application integration, using WebSphere Message Broker 8.0.0.1, integrating with EIS such as Siebel, SAP and CICS, as well as .Net and Worklight.

Part 1 describes the original, complex business case and how WebSphere Message Broker Version 8.0 makes the solution simpler and more modular. The video ends by demonstrating use of a mobile application to process a sales order through the integrated applications.

Part 2 demonstrates developing message flows in the IBM Integration Toolkit and then deploying them to the runtime. The video ends by demonstrating use of a mobile application to process a sales order through the integrated applications.

Integrating mobile applications with WebSphere Message Broker using IBM Worklight

An IBM developerWorks article that shows you how to use IBM Worklight and the Worklight server to develop hybrid mobile applications that invoke enterprise services deployed on WebSphere Message Broker.

Using the WebSphere Message Broker Mobile Service Pattern with WebSphere Cast Iron® Web API Management

An IBM developerWorks article that shows you how to use the Mobile Service pattern to quickly and easily take existing WebSphere Message Broker Version 8.0 services and expose them through WebSphere Cast Iron Web API Management.

.NET integration scenarios

IBM Integration Bus supports Microsoft and .NET integration, making it simple to integrate your Microsoft assets with the rest of your services and applications within small to mid-size or large departmental enterprises.

This topic links to a range of scenarios about using IBM Integration Bus for .NET integration. The scenarios are provided in various formats, including video and articles that are hosted on external sites.

Some scenarios, originally developed for WebSphere Message Broker, apply also to IBM Integration Bus.

Featured scenario: .NET service enablement for mobile applications

Background

Company A is a retail banking business that uses a Microsoft .NET application to work with accounts. The company wants to enable its customers to use mobile devices for their account functions. The company is looking to reuse the existing .NET code and class, wrapping them as a web service, and providing a mobile application.

The solution

Customers can use a mobile application that runs on a range of devices to access functions on their accounts; for example, to query the balance.

By using the IBM Integration Toolkit with the Worklight: Microsoft .NET request-response pattern that is provided, the application developers create a production-ready solution in a matter of minutes. The solution is based on a web service, which wraps around the existing .NET class, made available by an integration application that runs on IBM Integration Bus. The developers select the methods to be exposed in the web service. By using the pattern and .NET class, the toolkit builds a mobile application and the mobile integration logic for the Worklight platform.

Implementing the solution can be split into three parts:

Part 1 demonstrates the aspects that are related to the .NET service enablement. Parts 2 and 3 demonstrate the mobile integration that uses the .NET service.

For part 1, see the video * WebSphere Message Broker - Mobile Service Enablement Part1 (hosted on YouTube).

For the entire part 1 to part 3, see the full scenario that is outlined in "Featured scenario: .NET service enablement for mobile applications" on page 80.

More scenarios

A selection of more scenarios about using IBM Integration Bus for .NET integration.

WebSphere Message Broker V8 .NET integration
This video demonstrates how quick and easy it is to create a message flow that starts an existing .NET assembly on a Windows system, by using WebSphere Message Broker Version 8.0. With the easy-to-use tools provided, this can be completed in less than 5 minutes. Although the text-to-speech example is unlikely to be useful in an integration solution, this scenario shows just how easy it can be to invoke a .NET assembly that is already available.

Using Microsoft .NET in WebSphere Message Broker V8
This scenario, presented as a series of four tutorials on IBM developerWorks, demonstrates the support for Microsoft .NET that was new in WebSphere Message Broker Version 8.0.

Part 1: Using the .NETCompute node sample shows you how to use the .NETCompute node to filter, modify, and create messages, and provides a sample scenario along with explanatory C# code snippets.

Part 2: Integrating with Microsoft Word shows you how to create a simple web service definition by using a WSDL file that you drag onto a message flow. This flow can be called by a web service client by using SOAP over HTTP. The request message sent to the flow represents a sales order for various items from a store. When the message flow receives the data, it uses C# code within a .NETCompute node to create a short Microsoft Word document that contains the detail of the sales order in a table. The Word document is written to a directory, and then the message flow sends an acknowledgement message back to the web service client.

Part 3: Integrating with Microsoft Excel shows you how the IBM Integration Bus .NETCompute node can use the Open XML SDK API to interact with Microsoft Excel. A broker message flow receives an input XML file and records the hierarchy of the data into a Microsoft Excel spreadsheet. When the message flow receives and parses the data, it uses C# code within a .NETCompute node to update a Microsoft Excel spreadsheet.

Part 4: Using the .NETCompute node for exception handling shows you how to use the .NETCompute node for exception handling, by using examples that build in complexity as extra exception conditions are deliberately produced and then handled.

IBM WebSphere Message Broker and Microsoft .NET in Midmarket Solutions
An IBM Redbooks solution guide that describes the ability to use WebSphere Message Broker Version 8.0 to integrate Microsoft .NET applications into a broader connectivity solution.

This solution guide highlights two usage scenarios:

- Creating a single interface that can connect and integrate heterogeneous data stores for transforming data.
- Accessing a Windows Communication Foundation service from a message flow.

Mergers and acquisitions scenario

IBM Integration Bus manages the flow of information across two disparate IT infrastructures after a company merger.

This scenario describes how IBM Integration Bus is used by a fictitious insurance company to manage two disparate IT infrastructures after a small, Internet-based insurance company is acquired by a large, more traditional, insurance company. The description focuses on what happens when a potential customer requests a motor insurance quotation by using the merged company Web site. This scenario is based on a larger, more complex scenario that was published on developerWorks. To read the full scenario see the links at the end of the scenario description.

Background

Company A is a motor and general insurance company that has been in business for approximately 50 years and currently has approximately 5 million policyholders. The company uses agents and a call center to communicate with customers. The company has a large established IT infrastructure, which includes CICS Transaction Server for z/OS and IBM DB2 Universal Database™ on z/OS.

Company B is small Internet-based motor insurance company, which currently has less than 1 000 000 policyholders, and is expanding. The IT infrastructure managed by the company includes WebSphere Application Server on Microsoft Windows Server, and Oracle Enterprise and IBM DB2 Universal Database on Windows desktop.

The problems

Company A acquired Company B to gain access to the Internet-based insurance market and to use the Internet-based skills and IT infrastructure established in Company B. The two companies have customer and policy data of different formats but, for legal reasons, the data from the separate companies cannot be

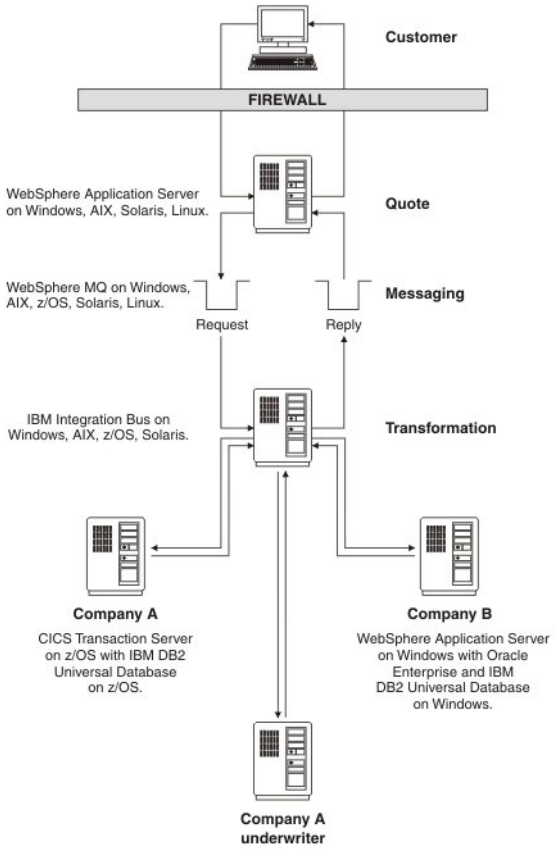
merged. However, the administration costs of managing the separate IT infrastructures are high. Also, customers, agents, and call center staff need a single administration process to interact with the company data.

The solution

Now that the two companies have merged, users can request an insurance quotation by giving some basic personal information in a form on the Web site of the new company. WebSphere Application Server, on which the Web site runs, forwards the request in XML format to IBM Integration Bus using the request queue in a WebSphere MQ cluster. IBM Integration Bus transforms the XML request to the COMMAREA format that is used by Company A systems, then routes the request to those systems. IBM Integration Bus also routes the request, in XML format, to Company B systems. Both systems return a quotation to IBM Integration Bus.

Logic within IBM Integration Bus also requests a risk assessment from the internal underwriter and applies the returned risk to the quotations from systems within Company A and Company B. The broker detects that, in this instance, the best or lowest quotation for the customer has been generated by Company A systems. Therefore, the broker transforms the quotation from Company A from COMMAREA to XML, and routes the quotation back to WebSphere Application Server to a reply queue in the WebSphere MQ cluster, where the quotation is stored for up to 14 days. WebSphere Application Server returns the quotation to the customer.

The following diagram shows the flow of information in this scenario.



Chapter 7. Samples

The IBM Integration Toolkit provides samples that show the features that are available in IBM Integration Bus, and how to use them. This topic provides links to the information about the individual samples.

Use the samples to learn how to use IBM Integration Bus.

- You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.
- Not all samples work in all modes. If you try to use a sample in a mode that is restricted, you receive a message indicating the reason; see Restrictions that apply in each operation mode. You can run all samples in a development and unit test broker; see “Development and unit test” on page 70.
- You might receive the following error:

  ```
  SEVERE: Could not find the connection files.
  ```

  If you receive this error, copy the `.broker` file for your default broker from the runtime workspace in which you created your default configuration to your current runtime workspace.
- To ensure that all available samples are displayed in the **Samples and Tutorials** tab in the IBM Integration Toolkit, in the "More samples" panel in the **Samples and Tutorials** tab click **Retrieve** and select the sample categories that you want to display.
-

  Note: A *message set* is the original container for message models used by IBM Integration Bus. In WebSphere Message Broker Version 8.0 and later, *message model schema* files contained in applications and libraries are the preferred way to model messages for most data formats. Message sets continue to be supported, and are required if you use the MRM or IDOC domains. If you need to model data formats for use in the MRM or IDOC domains, you must first enable message set development in the IBM Integration Toolkit. For more information, see Enabling message set development .

The samples are categorized as either Application or Technology samples.

Application samples
: The Application samples are small end-to-end IBM Integration Bus message flow applications that show how to transform and route messages through message flows.
  - Application samples

Technology samples
: The Technology samples are small IBM Integration Bus message flow applications that each show a specific feature of IBM Integration Bus.
  - Controlling and Routing samples
  - File Processing samples
  - Industry samples
  - Message Formats samples

- Message Transformation samples
- Monitoring samples
- Security samples
- Transports and Connectivity samples
- Web Service samples
- WebSphere ESB Conversion samples

Before you can use the samples, you must create the Default Configuration, see "Creating the Default Configuration" on page 96.

The following table lists the Application samples that are available in IBM Integration Bus.

Sample name	Description
Airline Reservations	This sample shows how to use a range of nodes, including nodes for aggregation, routing, tracing, filtering, and updating database tables.
Coordinated Request Reply	This sample contains two applications and one library. The library references a message set and broker project that are shared by both the applications. This sample demonstrates: • How two WebSphere MQ applications that have different message formats communicate with each other by using WebSphere MQ messages in a request-reply processing pattern, which is coordinated by using an MQGet node • How two JMS applications that have different message formats communicate with each other by using JMS messages in a request-reply processing pattern, which is coordinated by using a JMSReceive node
DatabaseInput Node	This sample demonstrates how to take data from a database, as it is being updated, and process the data within IBM Integration Bus.
Data Warehouse	This sample contains a message flow that archives data, such as sales data, to a database.
Error Handler	This sample contains a message flow and a subflow to show error handling in message flows.
Large Messaging	This sample shows how to process messages that contain repeating structures, and how to minimize the virtual storage requirements for the message flow.
Message Routing	This sample shows how to use a message flow to route messages to different WebSphere MQ queues based on data that is stored in a database table or a file.
Pager	This sample shows simple point-to-point messaging and publish/subscribe messaging. Use graphical interfaces to send text messages to a pager application, or to subscribe to publications about the surf on selected beaches.
Scribble	This sample is a small, graphical whiteboard application on which you draw by using your mouse pointer. Depending on the options you choose, you can see the effects of message transformation by using WebSphere MQ transport or RealTime transport.

Sample name	Description
Solar Pattern Authoring	This sample shows how to build an IBM Integration Bus pattern. The sample provides an example Integration project that calculates the sunrise and sunset times in a PHPCompute node. The sample also provides a pattern authoring project that configures a pattern.
User-defined Extension	These two samples show the use of user-defined nodes that are written in the C and Java programming languages.
Video Rental	This sample shows message transformation between three different formats: XML, Custom Wire Format (CWF), and Tagged/Delimited String (TDS) Format.
WBI JDBC Adapter Migration	This sample re-creates a scenario of migrating a JDBC adapter to invoke a message flow, by using an MQInput node with the built-in DatabaseInput node.
Web Service Aggregation	This sample demonstrates how to invoke a number of web services and amalgamate the results by using IBM Integration Bus aggregation nodes. The sample illustrates how aggregation can be used for transports other than WebSphere MQ, and highlights issues to be aware of. The sample also shows how you can use message flow monitoring to audit data across the aggregation fan-out and fan-in by using Collector node techniques.

The following tables list the Technology samples that are available in IBM Integration Bus.

Control and Routing samples

Sample name	Description
Aggregation	This sample shows how to use the Aggregation nodes to perform a basic four-way aggregation operation, with simple fan-out and fan-in message flows.
Business Rules	This sample demonstrates how to use business rules to apply decision logic, such as changing the discount offered to customers, based on changing business conditions.
Collector Node	This sample shows how to configure the Collector node to gather input from different input sources. It also shows some alternative methods for completing collections.
Simplified Database Routing	This sample shows how to use the following range of simplified (non-programming) message flow nodes: Route, DatabaseRoute, and DatabaseRetrieve. The sample illustrates how to access databases by using JDBC, and how to use values that are held in an acquired result set, gathered from a database query, to either dynamically route messages or update the content of the messages.
Timeout Processing	This sample shows how to use the timeout nodes to add timeouts to message flows.
Workload management	This sample demonstrates how an IBM Integration Bus administrator can control the rate at which message flows process data without redesigning the message flow.

File Processing samples

Sample name	Description
Batch Processing	This sample shows how to use the FileInput and FileOutput nodes to read different input files and append them to one output file. It also shows you how to read a file "as is" from one local input directory and write the file to a different local output directory.
File Output	This sample shows how a FileOutput node can write a message to a file during a message flow. The sample shows a flow updating a SOAP message, the FileOutput node writing this updated message to a file, and the message being sent back to the sender by using HTTP as the transport mechanism.
Managed File Transfer	This sample shows how the FTEOutput node writes a message to a file by using WebSphere MQ File Transfer Edition to manage the transfer of the file. The sample shows a Retail HQ to Branch product provisioning scenario. The flow receives a stream of WebSphere MQ messages with product data, and a file is created with an FTEOutput node. The built-in FTEAgent operates with the FTEOutput node to manage the transfer of the file to a remote location.
Connect:Direct File Transfer	This sample demonstrates how to use the CDInput and CDOutput nodes to work with IBM Sterling Connect:Direct in conjunction with IBM Integration Bus. The sample shows how to configure the IBM Sterling Connect:Direct nodes to send and receive files using IBM Sterling Connect:Direct.
WildcardMatch	This sample shows how to access the LocalEnvironment.Wildcard.WildcardMatch variable that is set in the FileInput node. The sample then gives an example of how, by using this variable, you can dynamically override the output file name and directory properties that are set in the FileOutput node.

Industry samples

Sample name	Description
Healthcare	This sample is a development accelerator, decreasing the time an integration developer within the healthcare industry requires to deliver integration solutions. This sample provides a number of Healthcare assets that solve key healthcare-specific integration problems.
TLOG Processor	This sample is a set of customized message sets, subflows, message flows, and style sheets that process transaction log (TLOG) data from retail stores. These samples provide sample TLOG input messages that are generated by various IBM Retail applications to test the TLOG message flows. Customers and service teams can customize or extend the TLOG Processor samples by modifying the message sets and flows, or by recombining these components in alternative ways.

Message Formats samples

Sample name	Description
COBOL	This sample shows how to model binary messages that are based on COBOL copybooks, by using DFDL.
Comma Separated Value (CSV)	This sample shows how to model common CSV message variants, and how to transform the sample CSV messages to and from XML.
EDIFACT	This sample shows an industry-standard message set for Edifact messages.
FIX	This sample shows an industry-standard message set for Fix messages.
ISO8583	This sample shows an industry-standard message model for ISO8583 messages using DFDL, and demonstrates how to transform ISO8583 messages received over TCP/IP.
SWIFT	This sample shows an industry-standard message set for Swift messages.
X12	This sample shows an industry-standard message set for X12 messages.
XMLNSC Namespaces	This sample shows how to change the existing namespace of a message, remove a namespace from a message, and add a namespace to a message.
XMLNSC Validation	This sample shows the capability of the XMLNSC parser to validate messages against an XML schema.

Message Transformation samples

Sample name	Description
Data Analysis	This sample demonstrates how to use the Data Analysis Toolkit to analyze XML data and create Data Analysis Tools.
JavaCompute Node	This sample shows how to use the JavaCompute node to perform tasks such as calling an external service and propagating a new message that is based on the results of the call.
Graphical Data Mapping Retail	This sample shows how to use a graphical data map on a Mapping node to transform messages and enrich them with data from a database, within a simple retail scenario.
.NETCompute Node	This sample shows how to filter, modify, and create messages by using the .NETCompute node in a message flow.
.NETInput node to MSMQ	This sample shows how to develop a .NETInput node to receive data and initiate a message flow from MSMQ by using .NET C# programming language.
PHPCompute Node	This sample shows how to use the PHPCompute node in a message flow to transform an XML message.
XSL Transform	This sample shows how to use a message flow to transform an XML message to another form of XML message according to the rules provided by an XSL stylesheet.

Sample name	Description
JavaCompute Node JAXB Transformation	This sample shows how to use the JavaCompute node to transform a message by using Java Architecture for XML Binding (JAXB).

Monitoring samples

Sample name	Description
WebSphere Business Monitor	This sample provides resources to help you use the monitoring events that are produced by your message flows for business process modeling by using WebSphere Business Monitor.

Security samples

Sample name	Description
Security Identity Propagation	This sample shows how to use Identity Security features to extract the security credentials from the messages on the MQInput and HTTPInput nodes. So that the sample can run stand-alone, the sample does not include security validation with an external security provider system, such as LDAP or TFIM. The sample also shows how to manipulate the security credentials by using ESQL, then how to propagate the identity to the MQOutput and HTTPRequest nodes.
Security Policy Enforcement Point (PEP)	This sample demonstrates how to use the SecurityPEP node as the Policy Enforcement Point in a message flow.

Transports and Connectivity samples

Sample name	Description
Browsing WebSphere MQ Queues	This sample shows how a message flow can browse WebSphere MQ messages that are in a queue, therefore retrieving the messages non-destructively. This sample also shows how to examine the contents of the browsed message to determine whether to get the message. Getting the message is a destructive process that removes the message from the queue.
CICS Transaction Server for z/OS Connectivity	This sample is based on a scenario in which a business wants to retrieve a record from a file resource on CICS Transaction Server for z/OS. The sample demonstrates how to use the CICSRequest node. With this node, you can run CICS applications, and retrieve data from CICS regions.
CICS Transaction Server for z/OS Channel Connectivity	This sample demonstrates how to call a channel-based CICS program. A CICS channel structure can be represented in IBM Integration Bus by a message collection. This sample demonstrates how to create and populate a message collection for the CICSRequest node and how to process the collection after the call.

Sample name	Description
CORBA nodes	This sample is based on a product warehouse scenario where a stock administrator wants to manage a stock control management system hosted on a CORBA server. The sample demonstrates how to use the CORBARequest node to invoke CORBA server applications.
Email	This sample consists of three message flows that show the use of sending and receiving emails. The emailform message flow provides an HTML input form to construct and submit an email message. The sendemail message flow receives the message and processes all the details that are associated with the email message. The recipients that are specified on the form receive the message as an email in the appropriate format, with any attachments. The getemail message flow processes the email that is sent and filters the email either to a WebSphere MQ queue, or saves the attachment to a file by using the FileOutput node.
HTTPHeader node	This sample consists of three message flows that show the different ways in which you can use an HTTPHeader node. The three message flows are: • Single WebService in MQ flow sample. This sample message flow shows how to create an interface between a WebSphere MQ application and web-based applications by using the HTTPHeader and MQHeader nodes. • Multiple WebService requests sample. This sample shows how to create and reset HTTP headers by using the HTTPHeader node. • Set Cookie HTTP reply sample. This sample shows how to add an HTTPReply header by using the HTTPHeader node in a request-reply session.
IMS Synchronous Request	This sample shows how to call an IBM Information Management System (IMS) transaction synchronously from within a message flow. The sample uses the IMSRequest node to make the synchronous calls by using IMS Connect. This sample uses the IMS sample transaction DSPALLI (Display All Invoices), which is typically available on all IMS systems. The DSPALLI transaction can call a REXX or COBOL program, although REXX is the default that is typically installed on IMS.
JD Edwards Connectivity	This sample consists of a message flow that demonstrates the use of the JDEdwardsRequest node. This sample message flow uses the JD Edwards business function call "retrieve" to fetch a record from a JD Edwards EnterpriseOne server. The record is then put on a WebSphere MQ queue.
JMS Nodes	This sample shows how to use the JMS nodes as a JMS Consumer and Producer to an external JMS provider.
JMSHeader node	This sample shows how to use the JMSHeader node in a JMS coordinated request-reply scenario.
MQHeader node	This message flow sample shows how to use the MQHeader node to add and remove an MQMD header.

Sample name	Description
SAP callout to a synchronous system	This sample consists of a single message flow that demonstrates the use of the SAPInput node with the SAPReply node to enable a message flow to act as a synchronous BAPI. The message flow is used to service requests for four different BAPIs that create, update, retrieve, and delete customer details.
SAP callout to an asynchronous system	This sample consists of three message flows that demonstrate the use of the SAPInput node with the SAPReply node to enable a message flow to act as a synchronous BAPI that wraps an asynchronous application. The message flows are used to service requests for four different BAPIs that create, update, retrieve, and delete customer details.
SAP Connectivity	This sample consists of two message flows that show the use of the SAPInput node and the SAPRequest node. The SAPInput node scenario shows how to use a message flow to receive IDocs from the SAP Material Master, then send the data to a WebSphere MQ output queue for processing by another message flow or application. The SAPRequest node scenario shows how to use a message flow to create a customer in SAP, then update and retrieve the customer details.
SCA nodes	This sample shows how to use the SCAInput, SCAReply, SCAAsyncRequest, and SCAAsyncResponse nodes to exchange message requests and responses with a business process in WebSphere Process Server. The sample re-creates a scenario in which a savings account is linked to a current account, and money can be transferred between the two accounts. In the outbound scenario, IBM Integration Bus passes money transfer requests to WebSphere Process Server, which hosts the savings account. This sample can be extended to include an inbound scenario where WebSphere Process Server passes requests onto IBM Integration Bus, which hosts the current account.
TCPIP Client Nodes	This sample consists of three message flows that show both synchronous and asynchronous communication from the IBM Integration Bus to a TCP/IP server. It also includes a simple message flow to simulate the TCP/IP server.
TCPIP Handshake	This sample shows how to implement an application-level handshake protocol for a synchronous request reply model of communication between a client and a server. The sample also includes two other message flows to emulate the client and server applications. You can replace these applications by external applications that use the same interfaces.
Twineball Example EIS Adapter	This sample shows how to use the WebSphere Adapter nodes by using the Twineball adapter, a self contained EIS, to synchronize a C system with an EIS.

Web Service samples

Sample name	Description
Address Book	This sample shows how to use the SOAPInput, SOAPReply, and SOAPRequest nodes to provide and consume a web service. Two sets of example input messages are provided: one set to call the consumer flow, which in turn calls the provider flow, and one set to call the provider flow directly. This sample can also be extended to show how to set up WS-Security for existing message flows for both a provider and a consumer.
Asynchronous Consumer	This sample shows how to use the asynchronous SOAP nodes when you call a web service. The web service simulates an order service, and the client shows how existing WebSphere MQ interfaces can be extended to make web service requests.
RESTful Web Service Using JSON	This sample shows how to front an existing service as a RESTful web service providing a JSON message format interface. The sample also shows how to consume the RESTful web service from a message flow.
SOAP Nodes	This sample shows the use of SOAP nodes to both provide and consume a web service.
Web services using HTTP nodes	This sample shows how to use IBM Integration Bus to front an existing application as a web service.
Web Services Gateway	This sample demonstrates how to use the SOAP nodes in a Web Services Gateway mode, which allows IBM Integration Bus to handle generic SOAP request/response and one-way messages when used as a web services provider or consumer.
WebSphere Service Registry and Repository Connectivity	This sample shows how to retrieve documents by using the WebSphere Service Registry and Repository nodes. You can use these nodes to query Service Registry information, and to use this information at run time. You can also use these nodes to acquire WSDL or other generic descriptions of available services.

WebSphere ESB Conversion samples

Sample name	Description
WebSphere ESB user-defined converters	This sample demonstrates how to convert WebSphere Enterprise Service Bus primitives and bindings by writing user-defined code for the WebSphere ESB conversion tool.

Creating the Default Configuration

You can create the Default Configuration of IBM Integration Bus by using the Default Configuration wizard. You can also remove the Default Configuration by using the link provided.

About this task

The Default Configuration wizard creates all the components that you need to import and deploy the IBM Integration Bus samples, and to build your own samples. For more information about the Default Configuration wizard and the authorities your user account must have to successfully create the Default Configuration, see "What the Default Configuration wizard creates."

You can use the following links only when you use the information center that is integrated with the IBM Integration Toolkit or the IBM Integration Explorer.

Procedure

Click the following link to start the Default Configuration wizard:

- Start the Default Configuration wizard

What to do next

When you have finished using the Default Configuration, you can remove it by clicking the following link:

- Remove the Default Configuration

What the Default Configuration wizard creates

A table of the components that are created by the wizard, details of how to resolve problems, and how to view errors.

Purpose

The components that are created by the Default Configuration wizard are listed in the following table.

Component	Name
integration node (broker)	IB9NODE
queue manager	IB9QMGR

- IBM Integration Bus automatically scans for a free queue manager port starting at 2414.
- The HTTP listener port that is automatically used is 7080.

Target environment of the wizard

The target environment contains the default components that are created by the Default Configuration wizard.

```
Target Environment of Wizard
-------------------------------------------------
Summarized resource updates listed below will be
applied to installation
[C:\Program Files\IBM\MQSI\9.0]
```

```
All actions are logged to file
[DefaultConfigurationWizard.log]
in the workspace directory
[C:\Documents and Settings\Administrator\IBM\IntegrationToolkit90\
workspace\.metadata]

All actions are applied under account:
LocalSystem
Queue manager name: IB9QMGR
Queue manager port: 2414

Default broker details
Broker name: IB9NODE
Queue manager name: IB9QMGR

HTTP listener port: 7080
```

Resolving problems when creating and running the Default Configuration

If you have problems when you run the Default Configuration wizard, consider whether the following issues apply to you:

- If you have any existing integration nodes that are using the default HTTP listener port of 7080, you must ensure that the existing integration nodes are stopped before you create and deploy to the Default Configuration. If a sample does not run as you expect, check the Event Log for errors such as BIP3144:

  ```
  BIP3144
  (IB9NODE.HTTPListener) An error has occurred during HTTP
  listener startup:
  the specified TCPIP port ('7080') is already in use.

  The HTTP listener needs to bind to a TCPIP port for correct
  operation to be possible.
  The broker-specific TCPIP port number '7080' is in use by
  another application.

  Stop other applications from using the specified port,
  or change the broker-specific port.
  ```

- Linux On Linux: Ensure that your user account belongs to the mqbrkrs group.

Viewing errors

To view errors generated by the Default Configuration wizard look at the `DefaultConfigurationWizard.log` file:

- Windows On Windows: This file is, by default, in `C:\Documents and settings\`*`user_name`*`\IBM\IntegrationToolkit90\workspace\.metadata\DefaultConfigurationWizard.log`
- Linux On Linux: This file is, by default, in `/home/`*`user_name`*`/IBM/IntegrationToolkit90/workspace/.metadata/DefaultConfigurationWizard.log`

To view Eclipse errors that are caused by the Default Configuration wizard look at the Eclipse Error log, see Viewing the Eclipse error log. The Error Log view opens in the perspective in which you are currently working.

Chapter 8. Glossary of terms and abbreviations

This glossary defines IBM Integration Bus terms and abbreviations that are used in this online information center.

A

adapter
An intermediary software component that allows two other software components to communicate with one another.

Administration API (CMP)
An application programming interface that your applications can use to control brokers through a remote interface. For continuity and consistency, this API is referred to as the CMP, its name in previous versions.

application
An object that functions as a virtual folder to organize shortcuts to other objects, external files, and URLs in a logical, job-specific, or project grouping.

attribute
A characteristic or trait of an entity that describes the entity; for example, the telephone number of an employee is one of the employee attributes.

An attribute may have a type, which indicates the range of information given by the attribute, and a value, which is within that range. In XML, for example, an attribute consists of a name-value pair within a tagged element, that modifies features of the element.

attribute group
A set of attributes that can appear in a complex type.

B

BAR file
See broker archive file.

binary large object (BLOB)
A block of bytes of data (for example, the body of a message) that has no discernible meaning, but is treated as one entity that cannot be interpreted.

BLOB See binary large object.

BLOB domain
The message domain that includes all messages with content that cannot be interpreted or subdivided into smaller sections of information. Messages in this domain are processed by the BLOB parser. See also DataObject domain, IDOC domain, JMS domain, MIME domain, MRM domain, SOAP domain, XML domain, XMLNS domain, and XMLNSC domain.

BLOB parser
A program that interprets a message that belongs to the BLOB domain, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output.

broker
A set of execution processes that host one or more message flows. Also known as a message broker and an integration node.

broker archive file
The unit of deployment to the broker; also known as a BAR file. The broker archive file contains a number of different files, including compiled message flows (`.cmf`) and message sets (`.dictionary` and `.xsdzip` files), that are used by the broker at run time. It can also contain additional user-provided files that your message flows might need at run time, if the file extension does not overlap with extensions that are used by the broker.

broker schema
A symbol space that defines the scope of uniqueness of the names of resources that are defined within it. The resources include message flows, ESQL files, and mapping files.

built-in node
A message flow node that is supplied by the product. Some of the supplied nodes provide basic processing such as input and output.

built-in pattern
A pattern that covers a set of commonly encountered message flow scenarios and that is packaged and released with IBM Integration Bus.

business component
This term is specific to the WebSphere Adapters. A component that defines the structure, behavior, and information that is displayed by a particular subject, such as a product, contact, or account, in Siebel Business Applications.

business object
A software entity that represents a business entity, such as an invoice. A business object includes persistent and nonpersistent attributes, actions that can be performed on the business object, and rules by which the business object is governed.

business rule
A constraint or required operation that applies to a specific set of business conditions or dependencies. An example of a business rule for a bank is that a credit check is not required when opening an account for an existing customer.

C

cardinality
See mapping cardinality.

certificate authority
A trusted third-party organization or company that issues the digital certificates. The certificate authority typically verifies the identity of the individuals who are granted the unique certificate.

cmf See compiled message flow.

compiled message flow (cmf)
A message flow that has been compiled to prepare it for deployment to the broker. A `cmf` file is sent to the broker within a BAR file.

complex element
A named structure that contains simple elements within the message. Complex elements can contain other complex elements, and can also contain groups. The content of a complex element is defined by a complex type. See also simple element.

complex type
A type that can contain elements, attributes, and groups organized into a hierarchy.

A complex type structure within a message contains elements, attributes, and groups organized into a hierarchy. See also simple type.

component directory
In z/OS, the root directory of the component's runtime environment.

component PDSE
In a z/OS environment, a PDSE that contains jobs to define resources to WebSphere MQ, DB2, and the broker started task. See partitioned data set.

connection
See message flow node connection.

content-based filter
In publish/subscribe, an expression that is included as part of a subscription to determine whether a publication message is received based on its content. The expression can include wildcards.

Custom Wire Format (CWF)
The physical representation of a message in the MRM domain that is composed of a number of fixed-format data structures or elements. In CWF, these structures are not separated by delimiter characters.

CWF See Custom Wire Format.

D

data element separation
For a complex type, defines to the MRM parser TDS physical format which method of identifying data elements is to be used, and how the data elements are constructed. The following separation types are supported: data pattern separation, delimited separation, fixed-length separation, and tagged separation

DataFlowEngine (DFE)
See execution group.

DataObject domain
The message domain that includes all messages that are exchanged between the broker and enterprise information system applications such as SAP, PeopleSoft, and Siebel. Messages in this domain are processed by the DataObject parser. You must create a message model for messages that you process in this domain. See also BLOB domain, IDOC domain, JMS domain, MIME domain, MRM domain, SOAP domain, XML domain, XMLNS domain, and XMLNSC domain.

DataObject parser
A program that interprets a message that belongs to the DataObject domain, and generates the corresponding tree from the business object on input, or the business object from the tree on output.

debugger
See flow debugger.

decision service
A collection of business rules that take one or more input and output parameters. See also: business rule.

destination list
See local environment.

digital certificate
An electronic document used to identify an individual, a system, a server, a company, or some other entity, and to associate a public key with the entity. A digital certificate is issued by a certificate authority and is digitally signed by that authority.

digital signature
Information that is encrypted with a private key and is appended to a message or object to assure the recipient of the authenticity and integrity of the message or object. The digital signature proves that the message or object was signed by the entity that owns, or has access to, the private key or shared-secret symmetric key.

Document Object Model (DOM)
A system in which a structured document, for example an XML file, is viewed as a tree of objects that can be programmatically accessed and updated.

document type definition (DTD)
The rules that specify the structure for a particular class of SGML or XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each item can be used within the particular class of documents. A DTD is analogous to a database schema in that the DTD completely describes the structure for a particular markup language.

DOM See Document Object Model.

DTD See document type definition.

E

EIS See Enterprise Information System.

element
A named piece of information, or a field, within a message, with a business meaning agreed by the applications that create and process the message. See also simple element and complex element.

embedded message
See multipart message.

EMD See Enterprise Metadata Discovery.

endpoint
A JCA application or other client consumer of an event from the enterprise information system.

Enterprise Information System (EIS)
The applications that comprise an existing enterprise system for handling company-wide information. An enterprise information system offers a well-defined set of services that are exposed as local or remote interfaces or both. (Sun)

Enterprise Metadata Discovery (EMD)
A specification that defines how you can examine an Enterprise Information System (EIS) and get details of business object data structures and APIs. An EMD stores the definitions as XML schemas by default, and builds components that can access the EIS.

environment
A structure within the message tree that is user-defined and can contain variable information that is associated with a message while it is being processed by a message flow.

ESM See external security manager.

ESQL See Extended SQL.

ESQL data type
A characteristic of an item of data that determines how that data is processed. ESQL supports six data types (Boolean, datetime, null, numeric, reference, and string). Data that is retrieved from a database or is defined in a message model is mapped to one of these basic ESQL types when it is processed in ESQL expressions.

ESQL field reference
A sequence of values, separated by periods, that identify a specific field (which might be a structure) within a message tree or a database table. An example of a field reference is `Body.Invoice.InvoiceNo`.

ESQL function
A single ESQL expression that calculates a resultant value from a number of specified input values. The function can take input parameters but has no output parameters; it returns to the caller the value that results from the implementation of the expression. The ESQL expression can be a compound expression such as BEGIN END.

ESQL module
A sequence of declarations that define MODULE-scope variables and their initialization, and a sequence of subroutine (function and procedure) declarations that define a specific behavior for a message flow node. A module must begin with the `CREATE node_type MODULE` statement and end with an `END MODULE` statement. The `node_type` must be one of Compute, Database, or Filter. The entry point of the ESQL code is the module scope procedure named MAIN.

ESQL procedure
A subroutine that has no return value. It can accept input parameters from and return output parameters to the caller.

ESQL variable
A local temporary field that is used to assist in the processing of a message.

event A change to a state, such as the completion or failure of an operation, business process, or human task, that can trigger a subsequent action, such as persisting the event data to a data repository or invoking another business process.

exception list
A list of exceptions, with supporting information, that has been generated during the processing of a message.

execution group
A named grouping of message flows that have been assigned to a broker. The broker enforces a degree of isolation between message flows in distinct execution groups by ensuring that they execute in separate address spaces, or as unique processes.

An execution group process is also known as a DataFlowEngine (DFE). This term is typically used in problem determination scenarios; for

example, trace contents or diagnostic messages). A DFE is created as an operating system process, and has a one-to-one relationship with the named execution group. If more than one message flow runs within an execution group, multiple threads are created within the DFE process.

An execution group is also known as an integration server.

exemplar
A project that contributes most of its content to a pattern. An exemplar contains message flows and other resources, such as source code.

Extended SQL (ESQL)
A specialized set of SQL functions and statements that are based on regular SQL, and extended with functions and statements that are unique to IBM Integration Bus.

Extensible Markup Language (XML)
A standard metalanguage for defining markup languages that is based on Standard Generalized Markup Language (SGML).

Extensible Stylesheet Language (XSL)
A language for specifying style sheets for XML documents. Extensible Stylesheet Language Transformation (XSLT) is used with XSL to describe how an XML document is transformed into another document.

External Security Manager (ESM)
In a z/OS environment, a security product that performs security checking on users and resources. RACF is an example of an ESM.

F

field reference
See ESQL field reference.

filter An ESQL expression that is applied to the content of a message to determine whether the message matches certain criteria.

For example, a Filter node uses a filter to determine how a message is to be processed. Alternatively, a filter is applied to the content of a publication message to determine whether it is to be passed to a subscriber.

flow debugger
A facility to debug message flows that is provided in the Debug perspective in the IBM Integration Toolkit.

GI

IDOC domain
The message domain that includes all messages that are exchanged between the broker and SAP R3 clients by the MQSeries® link for R/3. Messages in this domain are processed by the IDOC parser. See also BLOB domain, DataObject domain, JMS domain, MIME domain, MRM domain, SOAP domain, XML domain, XMLNS domain, and XMLNSC domain.

The IDOC domain is deprecated; use the MRM domain for new messages.

IDOC parser
A program that interprets a message that belongs to the IDOC domain, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output.

implementation function
A function written for a user-defined node or message parser; also known as a callback function.

inbound processing
The process by which changes to business information in an enterprise information system (EIS) are detected, processed, and delivered to a runtime environment by a JCA Adapter. An adapter can detect EIS changes by polling an event table or by using an event listener.

input node
A message flow node that represents a source of messages for a message flow or subflow. See also output node.

integration node
See broker.

integration server
See execution group.

J

Java Architecture for XML Binding (JAXB)
A Java binding technology that supports transformation between schema and Java objects, as well as between XML instance documents and Java object instances.

Java Database Connectivity (JDBC)
An industry standard for database-independent connectivity between the Java platform and a wide range of databases. The JDBC interface provides a call-level API for SQL-based and XQuery-based database access. See also Open Database Connectivity.

Java EE
See Java Platform, Enterprise Edition.

Java EE Connector Architecture (JCA)
A standard architecture for connecting the J2EE platform to heterogeneous enterprise information systems (EIS).

Java Message Service (JMS)
An application programming interface that provides Java language functions for handling messages.

Java Platform, Enterprise Edition (Java EE)
An environment for developing and deploying enterprise applications, defined by Sun Microsystems Inc. The Java EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multi-tiered, Web-based applications. (Sun)

Java virtual machine (JVM)
A software implementation of a processor that runs compiled Java code (applets and applications).

JAXB See Java Architecture for XML Binding (JAXB).

JCA See Java EE Connector Architecture.

JCL See Job Control Language

JDBC See Java Database Connectivity.

JMS See Java Message Service.

JMS domain
The message domain that includes all messages that are produced by the WebSphere MQ implementation of the Java Message Service standard. These messages, which have a message type of either JMSMap or JMSStream, are supported in the same way as messages in the XML domain, and are parsed by the XML parser. See also BLOB domain, DataObject domain, IDOC domain, MIME domain, MRM domain, SOAP domain, XML domain, XMLNS domain, and XMLNSC domain.

Job Control Language (JCL)
Job Control Language (JCL) comprises a set of Job Control Statements that are used to define work requests called jobs. JCL tells the operating system what program to run, and defines its inputs and outputs.

K

keystore
In security, a storage object, either a file or a hardware cryptographic card, where identities and private keys are stored, for authentication and encryption purposes. Some keystores also contain trusted, or public, keys.

L

library
In IBM Integration Bus, a project that is used for the development, version management, and organization of shared resources. A subset of the artifact types can be created and stored in a library, such as subflows, ESQL modules, message definitions, and Java utilities.

LIL See loadable implementation library.

loadable implementation library (LIL)
The implementation module for a node or parser written in C. This library file is implemented in the same way as a dynamic link library, but has a file extension of `.lil` not `.dll`.

local environment
A structure within the message tree that contains broker and, optionally, user information associated with a message while it is being processed by a message flow.

In previous releases, the local environment structure was known as the Destination list; the latter term is retained for compatibility.

local error log
A generic term that refers to the logs to which IBM Integration Bus writes records on the local system. Also known as the system log.

M

map (1) A complete transformation that has source objects that define the structure of the inputs and target objects that define the structure of the outputs. A map is represented as a `.msgmap` file.

(2) To associate a source to a target in a message map.

mapping
A target value expression.

mapping cardinality
The granularity of the way in which message elements are mapped from message source to message target. For example:

- One-to-one: associates a single source with a single target
- One-sided: associates a value with a target
- Many-to-one: associates multiple sources with a single target

message
Data that is passed from one application to another. Each message must have a structure and format that is compatible with both the sending and receiving applications.

message broker
See broker.

message definition
An annotated XML Schema model of a message format. A message definition is a structured collection of elements, types, and groups.

message definition file
A file in a message set that contains one or more message definitions.

message dictionary
A data structure that describes all the messages in a message set in a form suitable for use by the MRM parser.

message domain
A grouping of messages that share certain characteristics. A message domain has an associated parser that interprets messages that are received and generated by a broker. IBM Integration Bus supports messages in the BLOB domain, DataObject domain, IDOC domain, JMS domain, MIME domain, MRM domain, SOAP domain, XML domain, XMLNS domain, and XMLNSC domain. You can create additional parsers known as user-defined parsers to support messages that do not conform to the supported domains.

message flow
A sequence of processing steps that run in the broker when an input message is received. A message flow is created in the IBM Integration Toolkit by including a number of message flow nodes that each represents a set of actions that define a processing step. The connections in the flow determine which processing steps are carried out, in which order, and under which conditions. A message flow must include an input node that provides the source of the messages that are processed. Message flows are then ready to deploy to a broker for execution. See also subflow.

message flow node
A processing step in a message flow, also called a message processing node. A message flow node can be a built-in node, a user-defined node, or a subflow node.

message flow node connection
An entity that connects an output terminal of one message flow node to an input terminal of another. A message flow node connection represents the flow of control and data between two message flow nodes.

message format
The definition of the internal structure of a message, in terms of the fields and the order of those fields. When a message format is self-defining, the message is interpreted dynamically when it is read.

message group
A list of elements with information about how those elements can appear in a message. Message groups can be ordered, unordered, or selective.

message model
See message definition.

message parser
A program that interprets an incoming message and creates an internal representation of the message in a tree structure, and that regenerates a bit stream for an outgoing message from the internal representation.

message processing node
See message flow node.

message set
A folder in a message set project that contains one or more message definition files. It can be deployed to a broker in a broker archive file.

message set project
The eclipse container for a message set.

message tree
The logical tree structure that represents the content and structure of a message in the broker. The message tree is created by a message parser from the input message received by a message flow, according to a message template.

message type
The name given to a message definition in a message definition file.

metadata
The data that describes the characteristic of stored data.

MIME See Multipurpose Internet Mail Extensions.

MIME domain
The message domain that includes all messages that conform to the MIME standard. See also BLOB domain, DataObject domain, IDOC domain, JMS domain, MRM domain, SOAP domain, XML domain, XMLNS domain, and XMLNSC domain.

MIME parser
A program that interprets a message that belongs to the MIME domain, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output.

MQRFH
An architected message header that is used to provide metadata for the processing of a message. This header is supported by the WebSphere MQ (MQSeries) Publish/Subscribe SupportPac.

MQRFH2
An extended version of MQRFH, providing enhanced function in message processing.

MRM domain
The message domain that can parse, and write, a wide variety of message formats. This domain is primarily intended for non-XML message formats, but it can also parse and write XML messages. Message models are created in the IBM Integration Toolkit, with one or more physical formats. Messages in the MRM domain are processed by the MRM parser. See also BLOB domain, DataObject domain, IDOC domain, JMS domain, MIME domain, SOAP domain, XML domain, XMLNS domain, and XMLNSC domain.

MRM parser
A program that interprets a message that belongs to the MRM domain, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output. Its interpretation depends on the physical format that you have associated with the input or output message.

multipart message
A message that contains one or more other messages within its structure. The contained message is sometimes referred to as an embedded message.

Multipurpose Internet Mail Extensions
An Internet standard that defines different forms of data, including video, audio, or binary data, that can be attached to email without requiring translation into ASCII text.

N

namespace
In XML and XQuery, a uniform resource identifier (URI) that provides a unique name to associate with the element, attribute, and type definitions in an XML schema, or with the names of elements, attributes, types, functions, and errors in XQuery expressions.

XML instance documents, XML Schemas, and message definitions can use namespaces.

node (1) An endpoint or junction used in a message flow. See message flow node.

(2) Any element in a tree.

O

ODBC
See Open Database Connectivity.

Open Database Connectivity (ODBC)
A standard application programming interface (API) for accessing data in both relational and non-relational database management systems. By using this API, database applications can access data stored in database management systems on various computers, even if each database management system uses a different data storage format and programming interface.

operation mode
A property of a broker that determines what operations it can perform.

outbound processing
The process by which a calling client application uses the adapter to update or retrieve data in an enterprise information system (EIS). The adapter uses operations such as create, update, delete, and retrieve to process the request.

output node
A message flow node that represents a point at which messages leave the message flow or subflow. See also input node.

P

package group
A group of one or more packages that are designed to work together and can be installed to one directory. In IBM Integration Bus, this term refers to

the group of products that are installed and maintained by Installation Manager, which includes the IBM Integration Toolkit. Products that are installed into a package group share common files and resources. You can create multiple package groups on a single computer.

parser See message parser.

partitioned data set (PDS, PDSE)
In a z/OS environment, a data set in direct-access storage that is divided into partitions, which are called members. A partitioned data set (extended) (PDSE) is an extension to a PDS that contains an indexed directory in addition to the members.

pattern
A reusable solution that encapsulates a tested approach to solving a common architecture, design, or deployment task in a particular context.

pattern archive
An archive file that contains all the installable pattern resources. The pattern archive can be distributed by using a pattern community site.

pattern author
The developer that creates a pattern to meet a business or technical requirement.

pattern authoring
The process of configuring one or more regular projects to turn them into a pattern.

pattern authoring project
A project that contains the information used to create a pattern.

pattern categories
Categories that are based on pattern classification and that structure the display in the Patterns Explorer view.

pattern community site
An application that supports uploading, browsing, searching, and downloading pattern archives. The application can be either a website or a shared file system directory.

pattern instance
The implementation of a pattern, consisting of a pattern instance project and one or more regular IBM Integration Bus projects that implement the pattern. A pattern instance is generated by providing appropriate customization values to the parameters available in the pattern.

pattern instance project
A project that contains project references to all other projects in the workspace, relating to a specific pattern instance. A pattern instance project also contains a pattern instance configuration file that stores the pattern parameter values.

pattern parameter
A parameter that customizes and configures a pattern. For example, a queue name from which messages are read.

pattern user
A user who configures a pattern that the pattern author has created. The pattern is available to a pattern user in the Patterns Explorer view.

PDS, PDSE
See partitioned data set.

perspective
A group of views that show various aspects of the resources in the IBM Integration Toolkit. See also view.

physical format
The physical representation of a message within the bit stream. The supported physical formats are Custom Wire Format, XML Wire Format, and Tagged/Delimited String Format. Physical format information is used only by the MRM parser and the IDOC parser.

point-to-point
A style of messaging application in which the sending application knows the destination of the message. Contrast with publish/subscribe.

predefined element and message
An element or message for which a matching definition exists in the message model. See also self-defining element and message.

principal
An individual user ID (for example, a login ID) or a group. A group can contain individual user IDs and other groups, to the level of nesting that is supported by the underlying facility.

property
A characteristic of an object that describes the object. A property can be changed or modified. Properties can describe the name, type, value, or behavior of an object, and various other characteristics.

Resources that are created and maintained in the IBM Integration Toolkit and components have properties; for example, message flow nodes, deployed message flows, and brokers.

publication
In publish/subscribe messaging, a piece of information about a specified topic that is available to a queue manager, for delivery to subscribed applications.

publication node
An end point of a specific path through a message flow to which a client application subscribes, identified to the client by its subscription point.

publisher
An application that makes information about a specified topic available to a broker in a publish/subscribe system.

publish/subscribe
A style of messaging application in which the providers of information (publishers) are de-coupled from the consumers of that information (subscribers) using a broker. See also topic. Contrast with point-to-point messaging.

Q

QName
A name that conforms to the Namespaces in XML specification. A QName is a qualified name, which consists of an optional prefix, or its associated URI, and a local name.

queue A WebSphere MQ object to which message queuing applications can put messages, and from which message queuing applications can get messages.

queue manager
A system program that provides queuing services to applications. A queue manager provides an application programming interface (the MQI) that enables programs to access messages on the queues that the queue manager owns.

R

request/reply
A type of messaging application in which a request message is used to request a reply from another application. Contrast with datagram.

Resource Recovery Services (RRS)
A z/OS facility that provides two-phase sync point support by participating resource managers.

RRS See Resource Recovery Services.

S

SCA See Service Component Architecture (SCA).

schema
See XML Schema.

self-defining element and message
An element or message for which no matching definition exists in a message model, but which can be parsed without reference to a model. For example, a message that is coded in XML can be self-defining. See also predefined element and message.

send-and-forget
See datagram.

Service Component Architecture (SCA)
An architecture in which all elements of a business transaction, such as access to Web services, Enterprise Information System (EIS) service assets, business rules, workflows, databases, and so on, are represented in a service-oriented way.

simple element
A field in a message that is based on a simple type. A simple element can repeat, and it can define a default or a fixed value. See also complex element.

simple type
A characteristic of a simple element that defines the type of data within a message (for example, string, integer, or float). A simple type can have value constraints which place limits on the values of any simple elements based on that simple type. See also complex type.

SOAP A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet.

SOAP domain
The message domain that includes all messages that conform to the SOAP standard. You must create a message model for messages that you process in this domain. See also BLOB domain, DataObject domain, IDOC domain, JMS domain, MIME domain, MRM domain, XML domain, XMLNS domain, and XMLNSC domain.

SOAP parser
A program that interprets a message that belongs to the SOAP domain, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output. The bit stream is a representation of an XML file.

SQL See Structured Query Language.

SQLJ A Java extension that supports static Structured Query Language statements embedded within Java code.

Structured Query Language (SQL)
A standardized programming language that is used to define and manipulate data in a relational database. ESQL, the language that is used by IBM Integration Bus, is based on SQL, and has many similar constructs.

style sheet
A specification of formatting instructions that, when applied to structured information, provides a particular rendering of that information (for example, online or printed). Different style sheets can be applied to the same piece of structured information to produce different presentations of the information.

subflow
A sequence of processing steps, implemented by message flow nodes, that is designed to be embedded in a message flow or in another subflow. A subflow must include at least one Input or Output node. A subflow can be started by a broker only as part of the message flow in which it is embedded, and therefore cannot be deployed.

subflow node
A message flow node that represents a subflow.

subscriber
A publish/subscribe application that requests information about a topic.

substitution group
An XML Schema feature that provides a means of substituting one element for another in an XML message. A substitution group contains a list of global elements that can appear in place of another global element, called the head element.

system log
See local error log.

T

Tagged/Delimited String (TDS) Format
The physical representation of a message in the MRM domain that has a number of data elements separated by tags and delimiters.

target property
A message flow property that is selected by the pattern author to be configured by the pattern.

TDS Format
See Tagged/Delimited String Format.

terminal
The point at which one node in a message flow is connected to another

node. You can connect terminals to control the route that a message takes, dependent on the outcome of the operation that is performed on that message by the node.

topic A character string that describes the nature of the data that is published in a publish/subscribe system.

truststore
In security, a storage object, either a file or a hardware cryptographic card, where public keys are stored in the form of trusted certificates, for authentication purposes in Web transactions. In some applications, these trusted certificates are moved into the application keystore to be stored with the private keys.

type A characteristic of a message element that describes its data content. See also simple type and complex type.

U

uniform resource identifier (URI)
An encoded address that represents any resource, such as an HTML document, image, video clip, or program, on the Web; a URI is an abstract superclass compared with a Uniform resource locator or a Uniform resource name, which are concrete entities.

uniform resource locator (URL)
The unique address of an information resource that is accessible in a network such as the Internet. The URL includes the abbreviated name of the protocol used to access the information resource and the information used by the protocol to locate the information resource.

A Web server typically maps the request portion of the URL to a path and file name. Also known as universal resource locator.

uniform resource name (URN)
A name that uniquely identifies a Web service to a client.

URI See uniform resource identifier.

URL See uniform resource locator.

URN See uniform resource name.

user-defined extension
An optional component that is designed by the user to extend the functions of IBM Integration Bus. A user-defined extension can be either a node or a message parser. See also user-defined node and user-defined parser.

user-defined node
An extension to the broker that provides a new message flow node in addition to the nodes that are supplied with the product. See also implementation function and utility function.

user-defined parser
An extension to the broker that provides a new message parser in addition to the parsers that are supplied with the product. See also implementation function and utility function.

user-defined pattern
A pattern that is created by a pattern author.

utility function
A function provided by the broker that can be used by developers who write user-defined nodes or parsers.

V

value constraint
A limit that sets a restriction on the values that a simple type can represent.

view In Eclipse-based user interfaces, a pane that is outside the editor area, which can be used to look at or work with the resources in the IBM Integration Toolkit. For example, you can view and edit your project files in the Application Development view (previously called the Resource Navigator view). See also perspective.

W

Web service
A self-contained, self-describing modular application that can be published, discovered, and invoked over a network using standard network protocols. Typically, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available.

Web Services Description Language (WSDL)
An XML-based specification for describing networked services as a set of endpoints that operate on messages that contain either document-oriented or procedure-oriented information. A WSDL document enables a Web services client to invoke a Web service using the messages defined in a message definition.

WebSphere Adapters
WebSphere Adapters facilitate the exchange of business objects between Enterprise Information Systems (such as SAP Software, PeopleSoft Enterprise, and Siebel Business Application systems) and other applications.

IBM Integration Explorer
A graphical user interface based on the Eclipse platform for administering your brokers.

IBM Integration Bus pattern
A pattern in the IBM Integration Toolkit that exposes one or more pattern parameters for a pattern user to complete.

IBM Integration Toolkit
A graphical user interface built on Eclipse that is used to provide integration and connectivity solutions by developing resources associated with message flows.

WebSphere MQ Enterprise Transport
A transport protocol supported by IBM Integration Bus that enables WebSphere MQ application clients to connect to brokers.

WebSphere MQ Everyplace®
A generally available WebSphere MQ product that provides proven WebSphere MQ reliability and security for mobile and wireless devices. WebSphere MQ Everyplace applications connect to the broker using WebSphere MQ Mobile Transport.

WebSphere MQ Web Services Transport
A transport protocol supported by IBM Integration Bus that enables HTTP-compliant application clients to connect to brokers.

wildcard
A character that can be specified in subscriptions to match a range of topics.

work_path
The location in the local file system in which the component stores internal and working data.

working set
A logical collection of application projects, that you can use to limit the number of resources that are displayed in the Broker Application Development perspective.

World Wide Web Consortium (W3C)
An international industry consortium set up to develop common protocols to promote the evolution and interoperability of the World Wide Web.

WSDL
See Web Services Description Language.

W3C See World Wide Web Consortium.

X

XML See Extensible Markup Language.

XML domain
The message domain that includes all messages that conform to the W3C XML standard. Messages in this domain are processed by the XML parser. See also BLOB domain, DataObject domain, IDOC domain, JMS domain, MIME domain, MRM domain, SOAP domain, XMLNS domain, and XMLNSC domain.

The XML domain is deprecated; use the XMLNSC domain for new messages.

XML parser
A program that interprets a message that belongs to the XML domain and JMS domains, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output. The bit stream is a representation of an XML file.

XMLNS domain
An extension of the XML domain that contains messages that conform to the W3C XML standard, and that can also use the namespaces specification. Messages in this domain are processed by the XMLNS parser. See also BLOB domain, DataObject domain, IDOC domain, JMS domain, MIME domain, MRM domain, SOAP domain, XML domain, and XMLNSC domain.

XMLNS parser
A program that interprets a message that belongs to the XMLNS domain, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output. The bit stream is a representation of an XML file.

XMLNSC domain
An extension of the XML domain that provides high-performance XML

parsing and offers optional XML Schema validation. Messages in this domain are processed by the XMLNSC parser. You can create a message model for messages that you process in this domain, but a model is required only if you want to validate the message. See also BLOB domain, DataObject domain, IDOC domain, JMS domain, MIME domain, MRM domain, SOAP domain, XML domain, and XMLNS domain.

XMLNSC parser
A program that interprets a message that belongs to the XMLNSC domain, and generates the corresponding tree from the bit stream on input, or the bit stream from the tree on output. The bit stream is a representation of an XML file.

XML Path Language (XPath)
A language designed to uniquely identify or address parts of a source XML document, for use with XSLT. XPath provides basic facilities for the manipulation of strings, numbers, and Boolean values in message flow resources. For example, it can be used by Java programs within a JavaCompute node, or as the expression language within a Mapping node, or by the properties of several other nodes.

XML Schema
An international standard that defines a language for describing the structure of XML documents. An XML Schema formally describes and constrains the content of XML documents by indicating which elements are valid and in which combinations. (An XML Schema is an alternative to a document type definition (DTD), and can be used to extend functionality in the areas of data typing, inheritance, and presentation.) The XML Schema language is ideally suited to describing the messages that flow between business applications, and is widely used in the business community for this purpose. Message definitions are annotated XML Schema.

XML Schema Definition Language (XSD)
A language for describing XML files that contain XML Schema.

XML Wire Format
The physical representation of a message in the MRM domain that can be parsed as XML.

XPath See XML Path Language.

XSD See XML Schema Definition Language (XSD).

XSL See Extensible Stylesheet Language.

Chapter 9. Accessibility features for IBM Integration Bus

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Integration Bus. You can use screen-reader software to hear what is displayed on the screen.

- Supports keyboard-only operation
- Supports interfaces commonly used by screen readers

Tip: This information center, and its related publications, are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

If you are reading a PDF file with a screen reader, the default reading option typically returns the best results. In some cases, the way in which the PDF file has been generated might require you to select one of the other reading options; for example, `Use reading order in raw print stream`.

Keyboard navigation

This product uses standard Linux and Microsoft Windows navigation keys.

Visit the IBM Accessibility Center for more information about the commitments that IBM makes towards accessibility.

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 grant 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 106-0032, Japan

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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 Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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 measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

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

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

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information (www.ibm.com/legal/copytrade.shtml).

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44-1962-816151
  - From within the U.K., use 01962-816151
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Printed in USA