IBM Connectivity
Reviewer’s Guide

Enable connectivity infrastructure
for SOA solutions

Route and transform data
and messages

Discover, manage, and
govern services

Laura Gardash
Michael Hudson
Ben Mann
Anthony O’Dowd
Robert Phippen
Piotr Przybylski
Marc-Thomas Schmidt
Gari Singh

Luc Chamberland
John Colgrave
Dermot Flaherty
Greg Flurry

ibm.com/redbooks
First Edition (June 2008)

This edition applies to IBM WebSphere MQ, IBM WebSphere Message Broker, IBM WebSphere Enterprise Service Bus, IBM WebSphere DataPower, IBM WebSphere Adapters, IBM WebSphere Transformation Extender, IBM WebSphere Service Registry and Repository, and all subsequent releases and modifications until otherwise indicated in new editions.

This document created or updated on June 5, 2008.
# Contents

Notices .................................................................................................................. vii
Trademarks ........................................................................................................ viii

Preface .................................................................................................................... ix
The team that wrote this paper .............................................................................. ix
Become a published author .................................................................................. x
Comments welcome .............................................................................................. x

Chapter 1. SOA connectivity for enterprise solutions .............................................. 1
1.1 Introduction ....................................................................................................... 2
1.2 Typical connectivity scenarios ......................................................................... 2
1.3 Connectivity middleware building blocks ....................................................... 3
1.4 The IBM connectivity portfolio ....................................................................... 5
  1.4.1 WebSphere MQ ......................................................................................... 5
  1.4.2 ESB offerings from IBM WebSphere ...................................................... 5
  1.4.3 WebSphere Message Broker ................................................................. 5
  1.4.4 WebSphere Enterprise Service Bus ....................................................... 6
  1.4.5 WebSphere DataPower SOA Appliances ............................................. 6
  1.4.6 WebSphere Adapters ............................................................................ 6
  1.4.7 WebSphere Transformation Extender .................................................. 6
  1.4.8 WebSphere Service Registry and Repository ....................................... 7
  1.4.9 Integrating with the connectivity portfolio .......................................... 7
1.5 Summary ......................................................................................................... 7

Chapter 2. Enabling connectivity with WebSphere MQ .......................................... 9
2.1 Introduction ....................................................................................................... 10
  2.1.1 Understanding the basics ....................................................................... 10
  2.1.2 Programming model ............................................................................... 10
  2.1.3 Reliable business-data delivery ............................................................ 13
  2.1.4 Workload delivery .................................................................................. 13
  2.1.5 Easy administration .............................................................................. 14
2.2 Accelerating time to value ............................................................................... 14
  2.2.1 Overview ............................................................................................... 14
  2.2.2 Message Queuing Interface .................................................................. 14
2.3 Java Message Service ...................................................................................... 14
  2.3.1 Multi-language message service .......................................................... 15
  2.3.2 WebSphere MQ and z/OS ..................................................................... 15
2.4 Expanding the reach of connectivity ............................................................... 16
  2.4.1 Using a reliable SOAP transport ......................................................... 16
  2.4.2 Connecting Web 2.0 to core applications .......................................... 16
2.5 Service management and governance ............................................................. 17
2.6 Summary ......................................................................................................... 17

Chapter 3. Enabling connectivity with WebSphere Message Broker ...................... 19
3.1 Introduction ....................................................................................................... 20
3.2 Extending the reach of existing applications ................................................ 21
  3.2.1 Overview ............................................................................................... 21
  3.2.2 Broadening connectivity ....................................................................... 21
  3.2.3 Connecting Web services ..................................................................... 22
Chapter 4. Enabling connectivity with WebSphere Enterprise Service Bus

4.1 Introduction .................................................. 36
4.2 Accelerating time to value ................................. 37
  4.2.1 Overview ................................................. 37
  4.2.2 Invoking services .................................... 37
  4.2.3 Message splitting and aggregation ..................... 38
  4.2.4 Message transformation .............................. 41
  4.2.5 Refining message types ............................... 41
  4.2.6 Customizing capabilities ............................. 43
  4.2.7 Enhanced XSLT mapping editor......................... 44
4.3 Expanding the reach of connectivity ....................... 45
  4.3.1 Overview ................................................. 45
  4.3.2 WebSphere Transformation Extender integration ....... 46
  4.3.3 Supporting third-party JMS providers ................. 46
  4.3.4 Enhanced HTTP support ............................... 46
4.4 Summary ...................................................... 46

Chapter 5. Enabling connectivity with WebSphere DataPower ................. 47
5.1 Introduction .................................................. 48
5.2 Facilitating manageable, secure, and scalable SOA solutions .......... 48
  5.2.1 Overview ................................................. 48
  5.2.2 WebSphere DataPower Integration Appliance XI50 ......... 49
  5.2.3 WebSphere DataPower XML Security Gateway XS40 ....... 50
5.3 Accelerating time to value .................................. 51
  5.3.1 Overview ................................................. 51
  5.3.2 Configuration-driven approach to SOA ................. 52
  5.3.3 Enhanced standards support and interoperability ....... 52
  5.3.4 Integration with development and deployment environments . 53
5.4 Expanding the reach of connectivity ......................... 53
  5.4.1 Overview ................................................. 53
  5.4.2 Transforming messages ............................... 53
5.4.3 Content-based routing .................................................... 54
5.4.4 Protocol bridging ......................................................... 55
5.5 SOA management and governance ......................................... 57
  5.5.1 Policy-driven Web services management and SOA governance ........ 57
  5.5.2 Web service security and access control .............................. 57
  5.5.3 Web services policy ..................................................... 58
  5.5.4 Web services interoperability compliance ............................. 59
  5.5.5 SOA registry and repository support .................................. 60
  5.5.6 Web services management ............................................. 60
5.6 Summary ................................................................. 61

Chapter 6. Enabling connectivity with WebSphere Adapters .................. 63
6.1 Introduction ........................................................................ 64
6.2 Accelerating time to value .................................................... 64
  6.2.1 Overview ...................................................................... 64
  6.2.2 WebSphere Adapters portfolio ......................................... 65
  6.2.3 SAP Software ............................................................... 65
  6.2.4 Oracle E-Business .......................................................... 65
  6.2.5 Siebel Business Applications .......................................... 66
  6.2.6 JD Edwards EnterpriseOne .............................................. 66
  6.2.7 PeopleSoft ................................................................. 66
  6.2.8 E-mail ......................................................................... 67
  6.2.9 FTP ........................................................................... 67
  6.2.10 Flat Files ................................................................. 67
  6.2.11 JDBC ................................................................. 67
  6.2.12 CICS J2EE ECI resource adapter ..................................... 67
  6.2.13 IMS Connector for Java .............................................. 68
  6.2.14 Adapter delivery in WebSphere Integration Developer .......... 68
  6.2.15 Adapter tools ............................................................ 68
  6.2.16 Discovering services ................................................... 69
  6.2.17 Building services ....................................................... 70
6.3 Easier service creation ......................................................... 72
  6.3.1 Overview ...................................................................... 72
  6.3.2 Discovery setup ........................................................... 72
  6.3.3 Template-driven development ......................................... 72
  6.3.4 Automatic resource creation ......................................... 74
  6.3.5 Shared adapters ......................................................... 74
  6.3.6 Problem determination and monitoring ............................ 74
6.4 Extending the reach of adapters .............................................. 75
  6.4.1 Overview ...................................................................... 75
  6.4.2 WebSphere Message Broker ......................................... 75
  6.4.3 WebSphere Application Server ....................................... 76
  6.4.4 WebSphere Transformation Extender ............................... 76
  6.4.5 Building custom adapters ............................................. 76
6.5 Summary ........................................................................... 77

Chapter 7. Enabling universal data transformation with WebSphere Transformation Extender ........................................ 79
7.1 Introduction ........................................................................ 80
  7.1.1 Process complex data without manual coding ................... 80
  7.1.2 Service-oriented architecture ........................................ 80
  7.1.3 Mainframe scalability .................................................. 81
7.2 Understanding the basics ....................................................... 81
  7.2.1 Overview ...................................................................... 81
Notices

This information was developed for products and services offered in the U.S.A.

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

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

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

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.

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.

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.
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. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US 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 http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- AIX®
- CICS®
- DataPower®
- DB2®
- eServer™
- IBM®
- IMS™
- Parallel Sysplex®
- Rational®
- Redbooks®
- Redbooks (logo) ®
- SupportPac™
- System z™
- Tivoli®
- WebSphere®
- z/OS®
- zSeries®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

ABAP, BAPI, SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

EJB, Enterprise JavaBeans, eXchange, J2EE, Java, JavaBeans, JavaScript, JDBC, Streamline, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Visual Basic, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Enterprise IT environments continue to grow in complexity, through natural expansion or mergers and acquisitions. As the pace of business demand accelerates, IT environments look more for opportunities to flexibly reuse existing applications, services, and data. With the IBM® connectivity portfolio of products, these IT environments can discover and reuse services, expose and use application services, and route and transform messages. The set of capabilities (in the IBM connectivity portfolio) supports client requests for a broad service-oriented architecture (SOA) infrastructure that addresses the need to bridge and streamline communication in heterogeneous IT environments.

In this IBM Redpaper publication, we provide an overview of the IBM connectivity portfolio to market watchers who have a keen interest in understanding the most current connectivity technology releases, and how IBM is taking them to the next level. Specifically, we review the key benefits and features of the following products:

- IBM WebSphere® MQ
- IBM WebSphere Message Broker
- IBM WebSphere Enterprise Service Bus
- IBM WebSphere DataPower®
- IBM WebSphere Adapters
- IBM WebSphere Transformation Extender
- IBM WebSphere Service Registry and Repository

For more information about how the IBM connectivity portfolio facilitates application integration, see the following Web address:


The team that wrote this paper

This paper was produced by a team of specialists from around the world.

Luc Chamberland is the manager of Customer Enablement, WebSphere Business Process Management in IBM Canada.

John Colgrave is the WebSphere Service Registry and Repository (WSRR) Lead Architect and Universal Description Discovery and Integration (UDDI) Architect in IBM U.K.

Dermot Flaherty is the Messaging Products and Technologies Lead Architect in IBM U.K.

Greg Flurry is the SOA Advanced Technology Software Architect in IBM U.K.

Laura Gardash is a WebSphere products technical editor in IBM Canada.

Michael Hudson is a software architect for WebSphere Transformation Extender in IBM U.S.

Ben Mann is the WebSphere MQ Product Manager in IBM U.K.

Anthony O'Dowd is a software developer for WebSphere Message Broker in IBM U.K.

Robert Phippen is a software architect for WebSphere Enterprise Service Bus in IBM U.K.
Piotr Przybylski is a software architect for Service Component Architecture (SCA) and WebSphere Business Integration (WBI) Adapters in IBM U.S.

Marc-Thomas Schmidt is the Chief Architect Connectivity and Federated Enterprise Service Bus in IBM U.S.

Gari Singh is the SOA Appliances Product Manager in IBM U.S.

**Become a published author**

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at: [ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

**Comments welcome**

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  [ibm.com/redbooks](http://ibm.com/redbooks)
- Send your comments in an e-mail to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400
Chapter 1. SOA connectivity for enterprise solutions

This chapter introduces you to service-oriented architecture (SOA) connectivity for enterprise solutions. Along with an overview of SOA connectivity, this chapter provides details about the following topics:

- Typical connectivity scenarios
- Connectivity middleware building blocks
- The IBM connectivity portfolio
1.1 Introduction

Client information technology (IT) environments are typically broad in scope and heterogeneous by nature, performing various services across different parts of the organization. Many businesses have a diverse portfolio of applications that have been developed or acquired over a significant period of time, perform key business functions (ranging from order entry to business intelligence), and represent much of an enterprise’s intellectual capital and business advantage. The breadth of those applications is represented by the diversity of platforms, data formats, protocols, and interaction styles between applications and users. These interactions can vary from near real-time, online transaction processing to batch-oriented processing.

The need to react to new market opportunities increasingly requires information to flow between applications that were not designed to operate with each other. By creating new application connectivity and combining existing application assets, businesses can increase value by creating new applications by connecting existing applications. With intelligent application connectivity, you can reduce costs and increase revenue.

An SOA approach enables architectural flexibility and the reuse of services so that new and updated solutions, leveraging existing services, can be developed quickly, without the need for re-implementation. A fundamental challenge is that services are often implemented with different languages, expose different interfaces, use different programming models, and communicate using different protocols. Connectivity solutions provide the bridge for these environments to effectively communicate with one another.

Connectivity software provides the essential underpinnings for an SOA environment, enabling services to be accessible as solution building blocks and facilitating loosely coupled interactions between service requesters and service providers. Connectivity software enables the construction of composite business applications and business process management (BPM) processes, enriched by quality-of-service requirements and governance policies.

1.2 Typical connectivity scenarios

Typical connectivity scenarios include the following types:

- **Gateways**
  
  Expediting service requests between intra-enterprise and extra-enterprise applications. For example, an insurance broker might need to easily and securely connect to insurance providers. The intra-enterprise application (of the insurance broker) must understand and access services that are available from extra-enterprise service providers (insurance providers).

- **Internal connectivity**
  
  Enable disparate systems in an enterprise to make existing applications accessible for standards-based requesters to use. Connectivity solutions enable service requesters to access service providers through a wide range of standard-based protocols and manage necessary interface matching between the endpoints. In a mergers and acquisition situation where solution sharing is required among the merged entities, exposing existing services for reuse can be the quickest path to integration.
New business channels

Enterprises face a challenging task when considering the range of ways that business services can be reused. There can be daunting complexity in matching the spectrum of service requester access patterns and protocols (for example, Web browsers and Web services). The enterprise might find an equally broad range of access patterns, data formats, and protocols offered by existing applications (for example, MQ, CICS®, electronic data interchange (EDI), and FTP). Connectivity software supports the necessary protocol transcoding as well as the syntactic and semantic transformation of information. This range of service need is manifested, for example, in the case of multiple government agencies that are required to synchronize citizen records.

Web service interfaces

To use services in third-party systems (for example, packaged applications or enterprise information systems), you need access to the system through a Web service. For example, you can provide a Web service interface to an account management service on an SAP® system.

1.3 Connectivity middleware building blocks

The central architecture pattern of connectivity software is the enterprise service bus (ESB). An ESB acts as a logical intermediary in the interactions, provides loosely coupled interconnectivity between the requester of a function and the provider of that function. Its role as a logical intermediary allows the ESB to intercept messages and process them as they flow between a service requester and service provider. The ESB mediates between the two entities.

By being able to accommodate various types of mediation, the ESB can fulfill two core principles that support the separation of concerns between what general services are required versus how that is accomplished: service virtualization and aspect-oriented connectivity.

*Service virtualization* refers to the ability of the ESB to virtualize protocol and interaction patterns, the interface, and the location of the interacting participants. Interacting participants need not use the same communication protocol or interaction pattern. For example, a requester might require interaction through an inherently synchronous, remote procedure call (RPC) style of protocol. However, the service provider might require interaction by using an inherently one-way, event-, or messaging-based protocol, with two correlated interactions. The ESB provides the conversion that is needed to mask the protocol and pattern switch.

In addition, service requesters and providers do not need to agree on the interface for an interaction. For example, the requester might use one form of message to retrieve customer information and the provider might use another form. The ESB provides the transformation needed to reconcile the differences. A participant in an interaction does not need to know the identity (for example, the address) of the other participants in the interaction. For example, service requesters do not need to know that a request can be serviced by any of several potential providers at different physical locations. The actual provider is known only to the ESB and, in fact, might change without impacting the requester. The ESB provides the routing needed to hide the provider's identity.
With aspect-oriented connectivity, you can use the ESB to implement or enforce several quality-of-service capabilities of SOA solutions, such as security, management, logging, and auditing. Therefore, requesters and providers do not have to be concerned about these capabilities.

An ESB architecture can be manifested in different ways. A common approach is to have one logical ESB that connects all endpoints in the scope of an SOA. A common physical ESB implementation is a hub, with all service endpoints connected to the ESB. As the enterprise grows, the scope of the ESB also grows, from a relatively localized installation to an enterprise-spanning bus. Therefore, a federated approach is taken, leveraging a set of ESB domains that support the scope of service endpoint visibility and often reflect the organizational, governance, or simply geographical structure of the enterprise. There are a number of approaches to implement a federated ESB, from a set of peer domains to a hierarchically organized federation.

Implementations of the ESB patterns combine a number of building blocks that together deliver on the requirements of connectivity:

- A flexible messaging infrastructure that offers a spectrum of options for service endpoints to exchange information with the appropriate quality of interaction service
- Connectivity infrastructure that supports the core service virtualization and aspect-oriented characteristics of ESB connectivity
  It offers a framework for the mediation capabilities that is required to match service requesters to the ideal service provider counterparts and a set of predefined mediation functions that can be assembled into mediation scripts that enable the interactions.
- Connectivity infrastructure that is enabled by adapters at the edges of the ESB and supports a spectrum of interaction protocols and application programming interfaces (APIs) that are required to unleash the capabilities of existing and packaged applications in the form of service interactions in an SOA
- Transformation capabilities that support interaction where data format and structure are normalized to a canonical form, and that handle the transformation between what service requesters expect and what service providers can offer on-demand in the mediation layer
- Service endpoint governance that makes the capabilities and requirements of interaction endpoints and policies explicit as they govern interactions, matching service requesters and providers
  Service registries enable ESB participants to advertise endpoints and to govern their life cycle as well as the configuration of the ESB-facilitated interactions between the endpoints, enabling the ESB to make intelligent decisions to choose the correct endpoint based on the requirements of the service requesters.
- Service management that complements the core of the ESB capabilities, which facilitate interactions between endpoints
  Monitoring and managing those interactions, which includes instrumenting the connectivity infrastructure with agents that enforce monitoring or service-management policies, are important aspects of the overall connectivity experience.
1.4 The IBM connectivity portfolio

To support a broad range of connectivity scenarios, IBM provides several products, which are discussed in the following sections.

1.4.1 WebSphere MQ

WebSphere MQ offers various options for managing message exchange between service endpoints. WebSphere MQ has been a long-standing provider of reliable messaging with quality-of-service on interactions, ranging from guaranteed, once-and-only-once delivery to highly efficient best-effort delivery to low-latency messaging. It also offers quality-of-service enhancements to SOAP-based or Representational State Transfer (REST) interactions, or for interactions that require file-based access to business information. The universal messaging connectivity infrastructure integrates with all of the ESB products that fully use the quality-of-service options offered by WebSphere MQ.

1.4.2 ESB offerings from IBM WebSphere

IBM offers three enterprise service bus products:

- WebSphere Message Broker
- WebSphere Enterprise Service Bus
- WebSphere DataPower Integration Appliance XI50

Each product implements the basic capabilities of an ESB (protocol conversions, message transformation, enrichment, and routing) to further augment the processing of messages in flight between service requesters and providers. Each ESB also offers unique capabilities to suit your connectivity requirements:

- WebSphere Message Broker is built for universal connectivity and transformation across heterogeneous environments.
- WebSphere Enterprise Service Bus is optimized to work with WebSphere Application Server to provide an integrated SOA platform.
- WebSphere DataPower Integration Appliance XI50 provides hardware for simplified deployment and hardened security.

A benefit of having various interoperable ESB products that each focus on different use cases is that these products can be used together to implement federated ESBs where often one size does not fit all. Therefore, whether you need a bus that supports an edge-of-enterprise Web services gateway, flexible integration capabilities for important backbone applications, or managed ESB satellites surrounding the core integration ESB, IBM provides the ESB option for your requirements.

1.4.3 WebSphere Message Broker

WebSphere Message Broker offers universal any-to-any connectivity that supports a broad spectrum of message formats (from XML, industry standards, existing, byte arrays to comma separated lists), often adding ESB capability to existing messaging-based networks. Existing or packaged applications are made accessible as services in an SOA. WebSphere Message Broker exploits the reach and reliability of the WebSphere MQ messaging backbone, offers unique quality-of-service capabilities on z/OS®, and is optimized for high-volume processing.
1.4.4 WebSphere Enterprise Service Bus

WebSphere Enterprise Service Bus provides ESB capabilities for enabling SOA-based business processes and supporting BPM scenarios that are implemented with WebSphere Process Server. A natural fit for users with WebSphere Application Server skills, WebSphere ESB also federates ESBs that focus on standards-based interactions with XML, SOAP, Java™, and Web services. Broad SOA solutions are enabled through high quality-of-service features: reliability, scalability, and extensive transactional support.

1.4.5 WebSphere DataPower SOA Appliances

WebSphere DataPower Integration Appliance XI50 redefines the boundaries of middleware by offering a core set of ESB capabilities in the form of a hardware appliance, which is easy to install and configure. One key usage pattern for this ESB is in Web service gateway scenarios because the appliance is well suited for deployment in a network DMZ configuration. This ESB is optimized to bridge between leading standard protocols, including Web services, messaging, files, and database access, at wire speed.

1.4.6 WebSphere Adapters

The IBM connectivity portfolio includes a rich set of WebSphere adapters that enable interactions with applications and resources that are constructed using various programming models or interaction protocols. Adapters manage normalized access to application endpoint-specific data formats or APIs and a range of quality-of-service options. In addition, adapters provide service enablement for enterprise applications in terms of service virtualization, enabling applications to “hop on the bus” regardless of their specific interaction protocols.

Various adapters are available, ranging from technology adapters, which support APIs and protocols, such as JDBC™ and FTP, to application adapters, which provide sophisticated integration for packages such as SAP applications. The adapters also assist the ESBs as plug-ins at the edges of ESB mediation processing, facilitating input from adapted applications to be put on the bus or enabling the bus to deliver requests to an adapter-enabled endpoint.

1.4.7 WebSphere Transformation Extender

While the ESB products offer first-class support for message transformation, there are cases where complex information transformation of format, structure, and content is required. In response, WebSphere Transformation Extender offers an unmatched spectrum of universal transformation capabilities with industry content packs covering a broad range of standards-based data formats (for example, EDI dialects and HL7). WebSphere Transformation Extender can also extend third-party middleware products or act as a stand-alone solution for enhanced batch transformation processing.

WebSphere Transformation Extender integrates with each ESB product:

- It integrates with WebSphere Message Broker either as an on-ramp or off-ramp to mediation processing or as a transformation plug-in as part of the mediation processing.
- It integrates with WebSphere Enterprise Service Bus at the edges of a mediated interaction.
- It integrates with WebSphere DataPower SOA Appliances in terms of providing transformation scripts.
1.4.8 WebSphere Service Registry and Repository

WebSphere Service Registry and Repository offers a broad range of service repository capabilities that a connectivity infrastructure requires. It manages and governs a catalog of endpoints that can participate in ESB-facilitated interactions, including endpoints that are implemented by actual service providers as well as virtualized services that the ESB implements. The registry includes a wealth of metadata about services. It includes documents that declare service interfaces, messages, or applicable policies in a standardized format (for example, Web Services Description Language (WSDL), XML Schema Definitions (XSDs) and WS-Policy documents). It also includes semantic decorations of those endpoints, such as classifications or properties that offer information about the service version, ownership, and current performance characteristics.

WebSphere Service Registry and Repository supports the governance of service interaction configurations and the service life cycle, and it enables the IBM ESB products to dynamically select interaction endpoints based on the relevant context, contract, or content. The three ESB products, WebSphere Message Broker, WebSphere Enterprise Service Bus, and WebSphere DataPower Integration Appliance XI50, help you retrieve and evaluate registry content so that you can manage ESB mediation behavior. In addition, you can use registry access control and service metadata lifecycle management capabilities to drive controlled updates to the interaction policies underlying ESB-managed interactions, such as for rules and for selecting endpoints.

1.4.9 Integrating with the connectivity portfolio

The products in the IBM connectivity portfolio are used in a broad range of usage scenarios, from basic message routing and service invocation to serving as the connectivity backbone of dynamic supply chain solutions. For example, WebSphere ESB provides core ESB capabilities in WebSphere Process Server for BPM scenarios.

While the ESB product offerings support a basic level of interaction monitoring, Tivoli® Composite Application Manager for SOA provides advanced service monitoring and management capabilities, such as the instrumentation of service interactions to enforce performance monitoring policies. In addition, Tivoli Composite Application Manager for SOA can send service status data to WebSphere Service Registry and Repository to further inform the dynamic selection of service endpoints.

1.5 Summary

Customers have diverse and complex application environments that require enhanced integration in order to achieve the flexibility and reuse of a changing business demands. The IBM connectivity portfolio offers robust and proven messaging, ESB, adapter, transformation, and service registry technologies to connect solutions, applications, and services.
Enabling connectivity with WebSphere MQ

This chapter discusses how you can achieve robust connectivity with WebSphere MQ. WebSphere MQ enables you to perform the following tasks:

- Connect applications and services across commercial IT systems
- Reliably transport data of all types between endpoints
- Participate in transactional updates
- Achieve easy entry into service-oriented architecture (SOA) through an innate, loosely coupled programming model
- Use on-ramps ranging from MQI to the JMS V1.1 interface
- Create a messaging backbone supporting an enterprise service bus (ESB)
2.1 Introduction

WebSphere MQ helps you integrate J2EE™, CICS, IMS™, and .NET environments and interoperate seamlessly with the messaging resources of WebSphere Application Server to form the basis for your ESB. With integrated support for Web services, you can quickly bring reliability and traceability to SOAP interactions. By using Eclipse-based tools, you can remotely configure your entire WebSphere MQ network. Furthermore, WebSphere MQ implements the industry-standard Java Message Service (JMS) interface and provides a non-Java interface.

2.1.1 Understanding the basics

Messages and queues are the building blocks of a messaging backbone. WebSphere MQ enables applications and services to communicate by sending messages rather than by calling each other directly. After all, messages are simply strings of bytes that contain the data that you want to deliver from one application to another. There are two parts to a message: a header that describes the message, and the data itself, which can be text-based (for example XML), binary data, or a bit stream. The message header identifies the message with a unique message ID and contains other fields, such as message type, information about its origin, priority, expiration time, and the queue to send any replies to.

Each WebSphere MQ message can be up to 100 MB. Larger messages and files can be transported if they are segmented into smaller chunks, which WebSphere MQ does automatically. Receiving applications can choose to retrieve the larger messages segment by segment or when they have been recombined by WebSphere MQ. Alternatively, programming controls enable you to split messages into segments based on logical boundaries or the size of the buffer that is available to the receiving application.

Alternatively, you can reduce message traffic by grouping many small messages that are going to the same destination into larger WebSphere MQ messages. When the composite message arrives at its destination, WebSphere MQ disassembles the message, preserving the order in which these constituent messages are delivered.

Distribution lists are another way to reduce messaging traffic. When the same message is sent to multiple queues that the same queue manager owns, only a single copy of that message needs to be sent to the queue manager. The queue manager uses a distribution list to determine which of the queues that it owns needs a copy of that message and then acts accordingly. You can update the distribution lists at any time.

2.1.2 Programming model

By using the flexibility of the WebSphere MQ programming model, you can more easily move into SOA. The programming model is based on communication using message queues. The sending applications direct their messages to target queues, which can be local to the sending application or remote. The receiving applications retrieve the messages from those queues, enabling greater flexibility as you create application interaction patterns that can
range from pseudo-synchronous request-and-response styles to fire-and-forget styles (or both).

A request-response interaction style is just what the name suggests. To the requester, the application appears to be a completely synchronous invocation of a server application with a response from the server after execution. In fact, the application is implemented by two distinct, asynchronous, one-way message exchanges.

By using WebSphere MQ APIs, you can combine two flows to make a pseudo-synchronous pattern. Figure 2-1 shows how many WebSphere MQ clients use such an interaction pattern to support customer-facing interactions and achieve response times that, in turn, achieve their business requirements. Application A invokes CICS Transaction B.

![Figure 2-1](image)

*Figure 2-1  Contrasting request-reply and fire-and-forget patterns*
The fire-and-forget style applies to a one-way message exchange in which a response is not required. Figure 2-2 shows how, by combining the two patterns, you can use the additional processing capability to enable parallel processing, therefore, improving overall throughput.

A key point here is that, by using message queues, WebSphere MQ provides an innately loosely coupled model for application communication where receiving applications pull from a queue at their own pace, thus providing time independence. This model is also well suited to an application queue in which messages of a similar type (for example, those that require processing by the same application) are placed on the same queue. Furthermore, the processing programs do not need to wait for input but instead can be triggered when there is work to do.

The pull model and triggering facilities in WebSphere MQ provide a powerful and flexible alternative to more traditional push models. As a result, you avoid the typical issues you get with push models when the target is not available. For example, you might not have a message-ordering requirement because each message represents a separate piece of work, such as an order-entry update, that is independent of its position in the message queue. In this case, the pull model supports multiple instances of server transactions that process a particular queue to maximize throughput. Each message carries the information that is required to process it, including reply information if this processing stage is part of a request-and-response flow.
2.1.3 Reliable business-data delivery

WebSphere MQ provides reliable message delivery to help ensure that the business data is properly propagated between applications. This reliability is achieved using transactional techniques that ensure that message data is delivered only once. When there is an external transaction coordinator, WebSphere MQ ensures that MQ messages are delivered according to transactional requirements. It also ensures that the transactions involving WebSphere MQ and other resource managers can run as two-phase commit transactions (shown in Figure 2-3), providing full transactional integrity.

![Figure 2-3 Transactions using a two-phase commit](image)

Additionally, on some distributed platforms, WebSphere MQ provides an XA transaction coordinator function to achieve the same degree of integrity for transactional updates.

2.1.4 Workload delivery

WebSphere MQ offers two key facilities to help with workload delivery: clustering, which is a platform-neutral technique, and IBM Parallel Sysplex® shared queues, which is a capability that is unique to the z/OS environment.

WebSphere MQ clustering is a widely used technique for spreading message data seamlessly among multiple queue instances to support parallel processing. In this manner, cloned applications can each access their own partition of the message data so that, if the application or its processing environment fails, the other partitions are still available until the appropriate restart action is taken.

![If the application or its processing environment fails, the other partitions are still available](image)

For the highest possible availability of message data combined with horizontal scalability, sysplex shared queues provide message data that you can access from any environment on a sysplex. They are independent of application or LPAR failure. In addition, WebSphere MQ Peer Recovery automatically resolves in-flight message data errors.
2.1.5 Easy administration

With WebSphere MQ Explorer, you can remotely configure WebSphere MQ from Linux® x86 and Microsoft® Windows® environments, without requiring a local server or client. WebSphere MQ Explorer ships with the distributed product but can be used to remotely configure WebSphere MQ for z/OS queue managers. You can also configure WebSphere MQ resources across the network using intermediate queue managers.

WebSphere MQ Explorer is also extensible. By using the documented interfaces to WebSphere MQ objects and resources in the tool, you can create your own extensions that are tightly integrated with the WebSphere MQ Explorer console. You can configure Message Queue Interface (MQI), JMS, and publish and subscribe messaging all graphically.

Because WebSphere MQ Explorer is based on Eclipse, it can shell-share with other products, such as WebSphere Message Broker, to provide a single integrated console. Furthermore, it is customizable. You can add new tools and features as plug-ins that are integrated into the console.

2.2 Accelerating time to value

This section discusses various messaging technologies that can help accelerate time to value.

2.2.1 Overview

WebSphere MQ provides a consistent API, called the Message Queuing Interface, across all of its supported platforms and programming environments to help make your programs portable. Along with providing a standard interface, WebSphere MQ fully implements the industry-standard Java interface—JMS Version 1.1, including the publish-and-subscribe approach to directing message flow.

2.2.2 Message Queuing Interface

MQI is the original programming interface for WebSphere MQ. It provides a set of verbs that are consistent across platforms. Depending on the operating system you are using, you can use C, C++, Java, PL/I, COBOL, Visual Basic®, ActiveX/COM, Assembler, RPG, or TAL to program the MQI. Support for other languages, such as Perl, is available as downloads.

2.3 Java Message Service

JMS is an industry-standard programming interface for messaging based on J2EE. J2EE-compliant application servers, such as IBM WebSphere Application Server, provide the JMS interface and a messaging-services implementation. Many messaging products, including WebSphere MQ, support JMS.
WebSphere MQ supports JMS Version 1.1. JMS applications can be sent between messaging products that support JMS without modification, which means that you need only one set of programming skills to use a range of vendor products. JMS standardizes only the interface. It does not standardize the technology that is used for the underlying data delivery. Therefore, the wire protocol that JMS-compliant products use is proprietary and vendor specific, which means that JMS-compliant products do not interoperate. However, vendors often provide extensions to the JMS API.

The JMS implementations available with WebSphere MQ and WebSphere Application Server do interoperate. WebSphere MQ and WebSphere Application Server can exchange messages to form a combined messaging backbone where the transactions are preserved and publish and subscribe definitions can be shared. This means that a Java application hosted in WebSphere Application Server can use JMS to communicate to other applications that are connected to WebSphere MQ, either by JMS or by its MQI interface. IBM products that are based on WebSphere Application Server, including WebSphere ESB and WebSphere Process Server, can also connect to WebSphere MQ in this manner and can use native MQI calls to connect to WebSphere MQ.

### 2.3.1 Multi-language message service

The JMS interface requires that programmers have Java skills. However, although Java is in widespread use today, it is not always the preferred programming language, nor is it optimal for connecting to many non-Java environments. IBM has developed APIs that are consistent with JMS but are implemented in additional languages. Because the API comes in a range of languages, it is called a Multi-Language Message Service (XMS).

WebSphere MQ clients include the XMS interface. The supported languages include C, C++, and a fully managed client for .NET environments, which can be used with any .NET language, such as C#.

### 2.3.2 WebSphere MQ and z/OS

WebSphere MQ provides consistent functionality across its supported platforms to provide application portability. In addition, WebSphere MQ uses the unique features and strengths of the z/OS platform for IBM eServer™ zSeries®. To provide the highest availability, capacity, and performance for persistent and non-persistent messages, WebSphere MQ for z/OS uses Parallel Sysplex.

WebSphere MQ for z/OS also provides tight integration with IBM CICS and IBM IMS by using the IBM MQSeries-CICS Dynamic Program Link (DPL) bridge, the IBM MQSeries-CICS 3270 bridge, and IBM MQSeries-IMS bridge. This means that CICS and IMS transactions can get implicit access to WebSphere MQ message data using the bridge technology in addition to explicit access to WebSphere MQ message data using MQI. Therefore, CICS and IMS transactions that are written to expect their input from (and direct their output to) terminals can be driven by message data, which extends the reach and usability of these assets by connecting them to the data that is accessible through the WebSphere MQ network.
2.4 Expanding the reach of connectivity

This section discusses two ways to expand the reach of connectivity:

- Using a reliable SOAP transport
- Connecting Web 2.0 to core applications

2.4.1 Using a reliable SOAP transport

Web services standards define mechanisms for classifying, externalizing, finding, and invoking services. These services can range from operations residing in your existing, existing applications—that can be made available to the rest of your enterprise—to new J2EE components that are sitting on an application server. In addition, as Web services standards develop, such concepts Web services reliable messaging and Web services notification are emerging.

Web Services Description Language (WSDL) defines the standards for Web services. Universal Description, Discovery and Integration (UDDI) provides directory and search capabilities. SOAP is the data format that is used when your application communicates with a Web service. However, a predicated standard for data transmission does not exist. You can send your SOAP messages by using the transport that suits you. One common method is to send SOAP messages over HTTP or HTTPS.

If you want your SOAP-formatted data to be delivered with quality of service and the benefits of a messaging backbone, you need to use a messaging product for the data transfer. WebSphereMQ can send and receive SOAP data in a Web services implementation (most commonly in the J2EE environment), which is often called SOAP over JMS. This enables Web services to take advantage of the benefits of a messaging backbone.

WebSphereMQ Bridge for HTTP provides access for Web 2.0, Web services, and applications that do not require MQ clients. If you want to create a rich Web experience with Asynchronous JavaScript™ and XML (AJAX) and Representational State Transfer (REST), the Bridge for HTTP helps you access business data from core applications, present it to Web users, and collect data from Web applications.
The AJAX commands provide REST access to message data on WebSphere MQ queues and to topics for publishing and subscribing. The examples in Figure 2-4 show you how easy it is to access MQ by using AJAX commands.

<table>
<thead>
<tr>
<th>Sample AJAX code and URIs</th>
<th>resource = Queue</th>
<th>resource = Topic</th>
<th>WebSphere MQ API equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>this.HttpServletRequest.open(&quot;GET&quot;, &quot;<a href="http://host/msg/queue/resource">http://host/msg/queue/resource</a>&quot;);</td>
<td>Reads next message from Queue</td>
<td>Reads next message on that topic</td>
<td>MQGET with browse</td>
</tr>
<tr>
<td>this.HttpServletRequest.open(&quot;POST&quot;, &quot;<a href="http://host/msg/queue/resource">http://host/msg/queue/resource</a>&quot;);</td>
<td>Added data as message to Queue</td>
<td>Publishes data on that topic</td>
<td>MQPUT</td>
</tr>
<tr>
<td>this.HttpServletRequest.open(&quot;DELETE&quot;, &quot;<a href="http://host/msg/queue/resource">http://host/msg/queue/resource</a>&quot;);</td>
<td>Retrieves and removes next message from Queue</td>
<td>Reads subscription and deletes message</td>
<td>MQGET</td>
</tr>
</tbody>
</table>

Figure 2-4  Example AJAX commands for accessing WebSphere MQ

2.5 Service management and governance

The WebSphere MQ service definition helps you describe WebSphere MQ applications as SOA services. It is published as a SupportPac (MA93) and uses Internationalized Resource Identifiers (IRIs) and WSDL.

The IRI specification that the WebSphere MQ service definitions defines addresses for referring to WebSphere MQ queues and topics in a way that is similar to how you refer to URL addresses. It provides a WSDL binding specification that defines extensions for WSDL that describe WebSphere MQ applications, including its connections, the queues or topics it uses, its message exchange pattern (request-response or one-way), and the quality-of-service and message formats it uses. You can create WebSphere MQ service definitions for applications that use either SOAP or native (non-SOAP) messages.

With these specifications, you can write WebSphere MQ service definitions that help you catalog your applications in a standard, machine-readable way and develop, manage, and monitor WebSphere MQ applications.

2.6 Summary

By using the rich feature set of WebSphere MQ, applications that run on various platforms can access and process business data. Access to WebSphere MQ can be either explicit, by using APIs, or implicit, using bridge technology for the CICS and IMS application-processing environments. The innate, loosely coupled programming model provides a natural entry point into SOA and the recent provision of a service-definition facility, whereby WebSphere MQ applications can be classified in a standard way, further supports WebSphere MQ in SOA.
Enabling connectivity with WebSphere Message Broker

This chapter discusses how you can enable connectivity with WebSphere Message Broker. WebSphere Message Broker helps you perform the following tasks:

- Connect all of your applications regardless of transport mechanism or data format
- Synchronize information between your packaged applications
- Easily transform your batch-oriented file work into online requests
- Create secure connectivity architecture for your applications
- Connect your MQ infrastructure to non-MQ aware systems
- Control your enterprise service bus (ESB) behavior using a registry
3.1 Introduction

WebSphere Message Broker offers significant value by extending existing applications through new connectivity. This connectivity extends to packaged applications and file-based processing. Everything can be connected, so that batch-oriented and online processing can operate in a secure infrastructure that is fully integrated with the application life cycle. WebSphere Message Broker provides universal application connectivity within a flexible and dynamic infrastructure. From a connectivity designer’s perspective, it provides facilities to connect requests from a wide range of applications to a correspondingly large set of applications.

Specifically, WebSphere Message Broker supports various transport protocols. These protocols include MQ, Java Message Service (JMS) 1.1, HTTP and HTTPS, SOAP-based Web services (with specific support for advanced Web services standards such as WS-Addressing and WS-Security), file, packaged applications (such as SAP, Siebel®, and PeopleSoft®), raw TCP/IP sockets, and user-defined protocols. These transports and protocols require support for multiple data formats including binary data formats, such as those found in C and COBOL applications, text-based industry formats (such as SWIFT, EDI, HIPAA, HL7) and support for XML.

As requests that use different transports, protocols, and data formats pass through WebSphere Message Broker, you, as the connectivity designer, can perform different operations. Such operations include routing, filtering, transformation, enrichment, and correlation to effectively provide the required connectivity behavior of the requesting and consuming application.

From a connectivity designer’s perspective, there are three key development artifacts: the message flow, the processing nodes, and the message tree data structure. These artifacts are graphically modeled in an Eclipse development environment during the design phase and subsequently deployed to a run time infrastructure.

The message flow is a simple wiring diagram that depicts how applications are connected. The message flow is comprised of the processing nodes, each of which performs a specific connectivity function in the flow, such as accepting an input request, transforming a message, or accessing a database. By using the message tree data structure, connectivity designers can describe their request data in a format-independent manner using built-in parsers.

With data format-independence, designers do not have to be concerned with the details of a particular request when they connect applications together. Data format-independence also enables a broad range of processing nodes, including specific nodes that allow custom skill sets, such Java, extended SQL (ESQL), Extensible Stylesheet Language Transformation (XSLT), and WebSphere Transformation Extender, to act upon a request as the request passes through a message flow. There are also graphical mapping capabilities for non-programmers.

Do not be concerned with the details of a particular request when you connect applications.

From an operational perspective, WebSphere Message Broker artifacts can be deployed to various systems from Microsoft Windows to z/OS, and includes UNIX® and Linux on different hardware and software platforms. The run time is always integrated with the operating system platform to provide an operational experience consistent with the users of the platform. However, the run time also provides consistent behavior, enabling users to make a true platform choice between and within deployments. The WebSphere Message Broker run time provides a highly scalable environment and extensive administration and systems management facilities for developed solutions, giving operational staff far-reaching control over deployed resources.
3.2 Extending the reach of existing applications

This section discusses various ways you can use WebSphere Message Broker to increase the connectivity of your applications.

3.2.1 Overview

Existing applications can be revitalized if they use the connectivity options available in WebSphere Message Broker, which continues to introduce new transport and protocol connectivity options and significantly enhance its data formats support. This increases the number of existing application types that can be connected to applications by using more recently developed protocols and formats. Such an example is the ability to connect an existing MQ-based application that consumes a binary data structure to one that uses SOAP-based Web services (shown in Figure 3-1).

3.2.2 Broadening connectivity

By using Message Broker Toolkit, you can design message flows that, when deployed, connect different application requests and responses. Message Broker Toolkit graphically represents the many transport and protocol connectivity options in the node palette (shown in Figure 3-2).
This palette includes a large set of drawers, each containing nodes that support specific application connectivity transports and protocols, elegantly organized to provide ready access to over 70 nodes.

With WebSphere Adapters nodes, you can connect node applications to systems such as SAP, Siebel and PeopleSoft. There is also support for database applications, with both ODBC and JDBC connectivity options, as well as applications enabled through File and E-mail-based technologies are supported. For smaller devices, such as PDAs, and telemetry devices, such as sensors and actuators, nodes support the MQe and SCADA protocols, which handle periodically disconnected and smaller, limited footprint devices, respectively.

### 3.2.3 Connecting Web services

WebSphere Message Broker can act as an interface to Web service-enabled applications (inbound support) and integrate Web services with application interactions (outbound support). In the first scenario, one part of the application is Web services literate, whereas in the second scenario, Web services might be used to provide facilities to applications that are not Web service aware. By using Message Broker Toolkit, you can simply drag a Web Services Description Language (WSDL) file onto the canvas to automatically create a message flow skeleton for both inbound and outbound scenarios. Figure 3-3 illustrates a WSDL file that is used to expose a message flow as a Web service.

![Figure 3-3: Exposing a message flow as a Web service by using a WSDL file](image)
In addition, you can easily configure the higher-level protocols WS-Addressing and WS-Security using a policy-based approach rather than by explicitly programming them. By using the WebSphere Message Broker Policy Set Editor (shown in Figure 3-4), you can, for example, specify that inbound requests have the correct digital signature. Alternatively, you can specify that outbound requests have either the whole payload or specific fields properly encrypted so that only the intended recipient can view them.

![Figure 3-4  The WebSphere Message Broker policy set editor](image)

WebSphere Message Broker makes it straightforward to integrate Web services with existing applications and provide the facilities to simplify more complex aspects of standards such as WS-Security.

### 3.2.4 Using MQ infrastructures

WebSphere MQ provides first-class support for robust, transactional, reliable, high-performance messaging. Figure 3-5 shows the five different nodes that support MQ operations in message flows, including point-to-point operations, such as MQGET and MQPUT, and publish-subscribe operations that use the MQ transport. Figure 3-5 includes the full range of MQ operations that are possible by using the built-in broker nodes.

![Figure 3-5  Different nodes supporting MQ operations in message flows](image)
The MQInput and MQGet nodes also include WebSphere MQ browse functionality, which means that these nodes can exploit every flavor of MQ operations. With browse support, users can read the contents of a queue while leaving the messages in the queue and, by using a Timer node, develop repeatedly triggered messaging applications based on the queue contents. Figure 3-6 shows how to configure MQ browse support on the MQInput and MQGet nodes.

![Figure 3-6 Configuring MQ browse support on the MQInput node](image)

### 3.2.5 Integrating disparate JMS providers

WebSphere Message Broker supports any JMS 1.1-compliant provider. Therefore, it can bridge between different JMS providers while preserving transactions. In doing so, you can unify processing across your disparate JMS domains and share information ordinarily restricted to particular technology silos.

Usually, JMS providers cannot operate with each other because their wire protocols are incompatible. However, with WebSphere Message Broker, multiple providers can exchange point-to-point and publish-subscribe messages across and between different domains. Now you can bring together your JMS messaging from a range of JMS providers, as well as integrate your JMS messaging with all of the other transport and protocol options that WebSphere Message Broker provides, such as MQ, HTTP, HTTPS, Web services or file. For example, if you want to bridge your JMS provider to your MQ network, just use the MQ and JMS nodes. If you want to expose those JMS applications as Web services over a SOAP interface, use the SOAP and JMS nodes together.

WebSphere Message Broker also makes configuring the JMS nodes easy by including pre-built administrative configurations for the most popular IBM and non-IBM JMS providers.

---

*Bring together your JMS messaging from a range of JMS providers*
For example, in Figure 3-7, the JMSReply node sends a reply in response to a JMS input message.

![Figure 3-7 JMSReply node sending a reply in response to a JMS input message](image)

Use the MQ and JMS nodes with any of the other nodes on the palette.

### 3.2.6 Connecting devices and telemetry to the enterprise

Thanks to the MQe and SCADA nodes (shown in Figure 3-8), WebSphere Message Broker can interact with mobile devices and telemetry devices, such as sensors and actuators.

![Figure 3-8 MQE and SCADA nodes](image)

By providing support for the MQe and SCADA protocols and the disconnected and small footprint devices that generate and use them, equipment ranging from mobile computers, retail appliances, and mobile phones to devices used in more hostile environments, such as oil rigs, can integrate their data with the enterprise.

### 3.2.7 E-mail notifications from enterprise processing

Transport and protocol support is complete with WebSphere Message Broker support for e-mail generation with a dedicated e-mail node. You can generate applications that use e-mail input from other WebSphere Message Broker-connected applications, and you can use e-mail for notification (shown in Figure 3-9). Error situations inside message flows can now automatically generate an e-mail to inform one or multiple people. With support for e-mail attachments, you can include the content of the error message in the notification. You can also reconfigure the e-mail server address and authentication credentials so that redeployment is not required when e-mail configuration details change.

![Figure 3-9 Flow of e-mail notification](image)

This flow uses HTTP nodes to handle an HTTP request and send an e-mail, which is an effective and easy way to provide an HTTP interface to an e-mail server. The type of response
that the flow sends to the originating e-mail client depends on the response from the e-mail server.

### 3.2.8 Using databases in application connectivity

It is common in application connectivity scenarios for requests to be routed and augmented by data held in a relational database. To enable this support, WebSphere Message Broker provides several nodes (shown in Figure 3-10) that you can use to access various databases and their connectivity logic to enable database CRUD operations.

![WebSphere Message Broker nodes for accessing data](image)

For more advanced database operations, you can program the database connectivity nodes by using SQL. However, you can perform the most common patterns of routing and augmentation by using a point-and-click interface (and the DatabaseRetrieve and DatabaseRoute nodes), which removes the need for the connectivity designer to be a SQL specialist.

### 3.3 Using packaged applications

This section discusses increasing communication between your various packaged applications.

#### 3.3.1 Overview

In many enterprises, packaged applications such as SAP, Siebel, and PeopleSoft play a vital role for universal business process including purchasing, sales, inventory, and personnel management. Although packaged applications are powerful, they represent several challenges for connectivity.

These challenges stem from the focus on a specific aspect of the business, which keeps the processing and data separate from other packaged applications and applications that might use MQ, JMS, Web services, or even file-based interfaces for communication. Packaged application interfaces are often nonstandard such as Business Application Programming Interfaces (BAPIs) and IDOCs for SAP. As a result, packaged applications can have difficulty using and generating information both to and from other application types in a business.

However, WebSphere Message Broker solves this communication problem because it has SAP Input and SAP Request nodes, Siebel Input and Siebel Request nodes, and PeopleSoft Input and PeopleSoft Request nodes. These nodes connect these systems to other non-packaged and packaged applications within your enterprise. Now you can drive new work into your packaged application systems and get work from them to drive other systems, your applications integration, into a broader infrastructure.
3.3.2 Moving information to and from packaged applications

The SAP Input and SAP Request nodes (shown in Figure 3-11) are typical examples of packaged applications. By using the SAP Input node, you can register with SAP to use output from SAP and integrate that output within a message flow to initiate processing.

That message flow might take data from SAP and, using MQ, send it to an enterprise application running on CICS. Alternatively, the message flow might take information from SAP and publish it to a number of different subscribers using the JMS nodes so that everyone who is interested in SAP updates can be notified.

![Figure 3-11 SAP Input and SAP Request nodes](image)

The SAP Request node, with which a message flow can send information to SAP. For example, a message flow might expose SAP by using the Web services SOAP nodes so that users can perform SAP work by using a Web service, MQ, or JMS interface.

In many cases, SAP data needs to be transformed as it leaves or before it enters the SAP system so that it can be exchanged with existing applications without those applications knowing the data is in a format specific to SAP. When such a transformation is required, all of the existing WebSphere Message Broker features can be brought into play. Such features span its support for XML, binary, and text-related data streams, to its ability to transform data by using graphical mapping, SQL, Java, XSL, and WebSphere Transformation Extender facilities.

SAP Input and SAP Request nodes allow all applications to communicate with packaged SAP applications regardless of their connectivity transport, protocol, or data format.

3.3.3 Synchronizing different packaged applications

You often need to keep different packaged applications synchronized with each other. For example, an SAP system and a Siebel system might need to be kept up to date with each other as business processes take effect. There might be changes in SAP that need to be reflected in Siebel, or vice versa, such as a name and address change for an SAP customer being synchronized with the equivalent customer record in Siebel.

In response, you can build WebSphere Message Broker flows that keep packaged application systems synchronized with each other (shown in Figure 3-12). In this case, information from SAP is being sent to Siebel to be updated. It is also being distributed over JMS to interested subscribers.

![Figure 3-12 Synchronizing packaged applications](image)
WebSphere Message Broker facilitates communication between packaged applications and existing enterprise applications.

### 3.4 Combining file-based and online processing

This section discusses combining file-based and online processing.

#### 3.4.1 Overview

The use of files for information exchange between applications, both internally and externally, remains popular and effective. Although some processing is performed online, there are legitimate reasons, usually related to the way businesses run or physical processes occur, for using files to exchange information between applications.

For example, many businesses like to reduce unit transaction costs by aggregating requests over numerous clients before sending the requests to a third party. This kind of information exchange is handled using files. Furthermore, file-based information exchange is a fairly passive method of exchange between two enterprises because neither enterprise has to mandate the technology for the information exchange.

#### 3.4.2 Files containing first-class data

By using WebSphere Message Broker, you can process files in a manner consistent with the organization of the information in the file. For example, when you use the File input node to extract information from a file that starts a message flow (shown in Figure 3-13), you can think of the file as a single large block of data or, alternatively, as made up of many distinct records, each delimited from each other with particular separators. In this example, the information extracted from the input file is sent to an MQ queue, but it can just as easily be sent to an SAP system. Moreover, when the file is accessed, it can be read in its entirety and put to the queue as a single message or as a set of records each in its own MQ message. With this capability, you can use data that was static to drive online work.

Figure 3-13 shows how you can use File input and File output nodes and the filtering capabilities of WebSphere Message Broker to perform the processing that ad-hoc programs usually perform. In this example, a filter node analyzes an input file and high-value records are sent to a file that contains only high-value work. All of the other records are put into a different file. If you want to get this work online to an MQ-enabled application or into SAP, rather than into a file, you can also route the filtered work to an MQ queue or SAP by replacing the File output node with an MQ output node or SAP Request node, respectively.
3.4.3 Gigabyte files, record delimiters, storage, and performance challenges

The most notable difference between files and messages in service-oriented processing is the size of the data being exchanged. Although it is still rare for a service request or a message to exceed 10,000 bytes, files that are hundreds of megabytes and even gigabytes are increasingly common in business processes.

By using WebSphere Message Broker, you can effectively process very large files because WebSphere Message Broker provides comprehensive support for a range of delimiters that separate the file into records, such as DOS or UNIX Line Ends and custom delimiter (shown in Figure 3-15). However, WebSphere Message Broker steps beyond these simple facilities by using user-defined data, such as an XML schema or COBOL Copybook (after it is imported into a message set), to determine record boundaries. This capability also applies to outbound file processing, which means that files can be interpreted regardless of how they are composed of in records.

Furthermore, WebSphere Message Broker parsing technology requires only a few file records in storage at any one time to determine record boundaries, thereby vastly reducing the storage requirements when processing a large file. When processing a gigabyte-size file comprised of many records, WebSphere Message Broker requires little extra storage because it reads only the required records.

Because WebSphere Message Broker combines efficient record detection and efficient storage usage, you can process many thousands of records per second in a message flow on relatively modest hardware.
3.4.4 Accessing local and remote files

Files that WebSphere Message Broker processes can either reside on a locally attached file system or on an FTP server. The File input and File output nodes support both local and FTP-based access. When the transfer is from an FTP server to an instance of WebSphere Message Broker, the FTP input node contacts the FTP server to download the file to a local staging directory. When the transfer is complete, the file is processed in the same way that a local file is processed.

Outbound FTP processing is essentially the reverse process, involving the generation of a local file that is subsequently transferred to the FTP server by the File output node. The capabilities for comprehensive delimiter support and minimized storage usage are the same for local and FTP-based files. In addition, you can change FTP configurations, such as server location and security authorizations, without redeploying the message flow, making it easier for operational staff to manage the life cycle of a deployed FTP solution in WebSphere Message Broker.

By using the File input and File output nodes, you can integrate your file-based processing with all of your other enterprise applications, including packaged applications. The operational and performance characteristics of the file-processing capabilities allow many solutions that were based on ad-hoc user-written applications to be based on a single integrated technology base.

3.5 Participating in a secure infrastructure

This section discusses various methods that you can use to provide a secure infrastructure for your IT environment.

3.5.1 Overview

WebSphere Message Broker forms an important part of the security infrastructure in its role as a policy enforcement point (PEP). It performs various security functions including identity management, authentication, and authorization by using Policy Decision Point (PDP) technology, specifically Lightweight Directory Access Protocol (LDAP) or Tivoli Federated Identity Manager.

As work requests pass between applications, it is highly likely that the security domain of the requestor and provider are different. This is especially likely in application connectivity scenarios when the objective is to integrate applications that were not designed to operate with each other. For example, an application can generate a request by a particular user that can be for a service in which this user's identity might or might not be authorized. When the request arrives at the provider, the provider must authenticate the user's identity to ensure that the request has indeed come from the specified party. (In the case of a user name, the provider authenticates the user's identity with the user's password.) When the user has been authenticated, security processing authorizes the requestor to access the service.

Identity, authorization, and authentication are essential to providing proper participation in a secure infrastructure. They are each are supported in WebSphere Message Broker for MQ, JMS, HTTP, HTTPS, and Web service protocols.
3.5.2 Securing applications: Identity, authorization, and authentication

In WebSphere Message Broker, the MQ, JMS, HTTP, HTTPS and SOAP nodes can all extract an identity from an inbound request. In Figure 3-16, MQ nodes extract the identity from $Root.MQMD.UserIdentifier by default. Similarly, other transport nodes can extract identity from the appropriate default location for a JMS user ID, HTTP and HTTPS, and the WS-Security header. Alternatively, you can specify a user-defined location from where the identity can be extracted, which is useful when the identity, such as an organization name or department code, is buried in the request.

![MQInput node properties](image)

Figure 3-16 MQInput node properties
If you have configured a PDP, such as LDAP or Tivoli Federated Identity Manager, and the user's identity has been determined, the identity can be passed to the PDP for authentication and authorization. This delegation is important because it enables enterprise-wide security models to be implemented, which security policies use to determine resource access. When using Tivoli Federated Identity Manager, it is also possible to map the identity so that a new identity is generated from the input identity. This capability is useful when the identity needs to change as a request passes from a requestor to a provider. To achieve this configuration, you assign a security profile to the message flow (shown in Figure 3-17) which locates a PDP for security processing.

![Figure 3-17 Assigning a security profile to a message flow](image)

When security processing is complete, the rest of the message flow handles the request. The flow designer can handle security exceptions as part of the connectivity logic, for example, to generate an e-mail that notifies someone about a security error or routes a failed request for special handling.

The identity that is extracted from the request is available for processing, and the message flow designer can base connectivity logic around this identity. When a request is passed to an outbound transport node, the identity in the request populates the outbound request, thereby establishing the correct identity that the target service is to use.
3.6 Integrating with the application life cycle using a registry

This section discusses IBM products that catalog your application-related assets.

3.6.1 Overview

Increasingly, enterprises are cataloging their application and service assets in a WebSphere Service Registry and Repository to establish a reference of their application estate. In the same way as a business takes an inventory of physical assets, software application registries allow businesses to take an inventory of their key enterprise application assets.

The act of cataloging application assets classifies them and the relationships between them. For example, applications might be classified by their function, owning department, or a myriad of other schemes. Relationships between applications can be established to show dependencies. The person who catalogs the applications in a registry can also associate attributes with those applications to assist processing. For example, applications might have a Class=GOLD or Class=SILVER attribute that people who have access to the registry and determine how the information is processed can use to provide a different level of service according to customer status. These application attributes are often called metadata.

Application assets are supporting a more formal definition of their behavior, which makes them ideally suited to be stored in a registry. The most common example is WSDL. However, recently WebSphere MQ introduced an MQ application service definition. This definition allows formal information about MQ application assets to be held in a registry, such as the location of the application in the form of a queue and queue manager location in an MQ network, the message properties of the request or response, and the information-exchange pattern.

From a connectivity perspective, a registry gives an enterprise a huge advantage in terms of governance, specifically application versioning and decommissioning. As more products become enabled to access registries and determine their processing according to the contents of the registry, it is possible to make configuration changes without redeploying message flows. For example, if you change the location of an application, you can update the registry, and requests can be automatically rerouted by registry-enabled products, such as WebSphere Service Registry and Repository.

3.6.2 Application lifecycle connectivity

WebSphere Message Broker supports WebSphere Service Registry and Repository, which is an IBM runtime registry for application and services. It includes classification, relationships, and metadata, with two nodes (shown in Figure 3-18) that allow message flow designers to access WebSphere Service Registry and Repository as part of their connectivity logic at runtime.

![Figure 3-18   Nodes that you use with WebSphere Service Registry and Repository](image)

By using the first node, the RegistryLookup node, you can return application and service information in WebSphere Service Registry and Repository as-is to the requestor who can then use it for processing. This information is cached locally for high-performance access,
and a built-in mechanism synchronizes the cache with changes to the registry. The node can either retrieve information based on static classifications, relationships, and attributes, or derive classifications, relationships and attributes from the inbound request so that you can retrieve content that is specific to a particular request. The most common retrievals are for Web service and MQ-based service definitions, but users have been known to extract XSLT style sheets and performance-dynamic transformations based on the requirements of the target application.

By using the second node, the EndpointLookup node, you can retrieve content in the same way as the RegistryLookup node but automatically set target location information that subsequent transport nodes can use. For example, specific high-value customers can be forwarded to the Class=GOLD application services rather than the standard service, causing the EndpointLookup node to populate the Web service transport information to be ready for processing by a subsequent SOAP request node.

3.7 Summary

WebSphere Message Broker provides a valuable range of options that enable you to connect new and existing applications quickly and efficiently. These applications can use a broad range of transport and protocol options, with data formats from a similarly diverse spectrum. Now, you can connect all of your applications, whether they are MQ, JMS, HTTP, Web service, file-based, SAP, Siebel, or PeopleSoft, within a consistent and secure framework so that you can create new applications targeted at revenue-generating opportunities.
Enabling connectivity with WebSphere Enterprise Service Bus

This chapter discusses how you can enable connectivity with WebSphere Enterprise Service Bus. WebSphere Enterprise Service Bus helps you perform the following tasks:

- Integrate disparate services
- Add a new service without changing the client
- Combine data from different services
- Transform data easily by using visual tools
- Reliably route failed service request calls
- Create mediations quickly by using prebuilt functions
4.1 Introduction

In a service-oriented architecture (SOA), the enterprise service bus (ESB) is the connectivity layer that effectively enables any-to-any service connectivity, regardless of service location, the protocol used to communicate, the message formats they use to express the data that they process, or the quality of service that they expect. This means that you can move or completely re-implement service providers without impacting your service consumers. It also means that, when making new services available, the new service provider does not have to be aware of the different mechanisms that service consumers use to access the service.

WebSphere ESB provides a broad range of capabilities for enabling and managing service virtualization as a flexible intermediary between service requesters and service providers, providing the following capabilities:

- Virtualizing location through routing, which locates a service provider without requiring the service requester to know the provider's location
- Virtualizing communication protocols by supporting a rich set of communication protocols so that the service requester can use the capabilities of a service without requiring the service and the requester to use the same communication protocol
- Virtualizing physical message formats by supporting industry-standard message formats, such as XML, thereby removing the need for a dependency between service requester and provider on the message formats that they expect to use
- Being message structure independent by supporting message transformations so that the message can be restructured to match the expectations of the service
- Augmenting quality of service by supporting requirements, such as transactionality, security, and logging

WebSphere ESB includes mediation primitives that encapsulate individual intermediary capabilities, such as routing and transformation, and transport bindings that enable communication over various communication protocols. You can use WebSphere Integration Developer to assemble these primitives and bindings to create customized service intermediaries that enable communication between service requesters and service providers, without either needing to be changed. As an example, in response to an intermediary needing to route requests to different service providers based on message content, such as a requester classification, you visually assemble and configure the supplied functions in WebSphere ESB.

WebSphere ESB is built on WebSphere Application Server Network Deployment to use the scalability, clustering, failover, and security capabilities that the base server provides. In addition, WebSphere Process Server provides all of the functions of WebSphere ESB and adds functions for choreographing business processes and managing business rules.
Chapter 4. Enabling connectivity with WebSphere Enterprise Service Bus

4.2 Accelerating time to value

This section discusses features in WebSphere ESB that accelerate time to value.

4.2.1 Overview

The WebSphere ESB programming model accelerates the task of integrating services by using the Service Component Architecture (SCA). This means that in WebSphere ESB all elements of a business solution, such as access to Web services, enterprise information system (EIS) assets, and messaging applications, are represented in a service-oriented way.

For example, the bindings for SCA imports and exports perform protocol virtualization and data bindings perform message format virtualization. You can perform all other virtualization tasks by using the mediation flow programming model, which is a dataflow programming model that operates on an abstract representation of the service invocations and responses that flow through the ESB.

Message and protocol virtualization provide a common access model that is independent of the underlying transport and message-format implementations. A mediation flow is composed of mediation primitives, and each primitive transforms, routes, enriches, or logs information in a database.

4.2.2 Invoking services

By using the Service Invoke primitive, you can invoke services during either a request or a response flow, which is useful, for example, when you need to query a subsidiary service for information when the subsidiary service acts as an intermediary between requesters and providers. The Service Invoke primitive supports both synchronous and asynchronous invocation and both one-way (input-only) and two-way (request-response) services, enabling you to call services in the same way as a standard callout operation but from anywhere in the flow. The Service Invoke primitive (shown in Figure 4-2 on page 38) also provides out
terminals for each fault message as defined by the target service, which enables you to handle any message type that a service returns. Figure 4-2 shows the tools used to define a Service Invoke mediation primitive.

![Diagram of Service Invoke and tools](image)

**Figure 4-2** A Service Invoke primitive and the tools used to define a Service Invoke mediation primitive

As with the Callout mediation primitive, you can configure the Service Invoke primitive to invoke a statically specified service (defined using an import) or a service that is specified dynamically using an endpoint in the Service Message Object header. The advantage of the dynamic approach is the flexibility it provides. You can look up service endpoints that are defined in a service registry, a database, or other location at run time rather than by specifying a static location.

In addition, both the Service Invoke and the Callout mediation primitive retry a service multiple times. You can even use an alternative list of dynamic endpoints to call the service if the initial invocation fails.

### 4.2.3 Message splitting and aggregation

The Fan-Out and Fan-In primitives help you split and aggregate messages. You use the Fan-Out primitive in two ways: to explicitly split a message based on a repeating element in the message (called the iterate mode) or to act as a logical marker to allow a flow to be split and later rejoined at a Fan-In primitive. In the iterate mode, the output terminal of the Fan-Out primitive fires multiple times, once per instance of the repeating element in the message. For example, if the Fan-Out primitive is configured to fire for all repeated customer elements and receives the message (shown in Example 4-1) represented in XML syntax, it fires its output terminal four times, once for each occurrence of the customer element.

**Example 4-1 CustomerData with four customer elements**

```xml
<CustomerData>
  <Customer>
    <CustomerID>ABCD1234</CustomerID>
    <CustomerName>John Smith</CustomerName>
    <CustomerDOB>01/01/1942</CustomerDOB>
  </Customer>
</CustomerData>
```
To demonstrate the iterate function of the Fan-Out primitive, we consider the following scenario. A customer service processes the message structure in the syntax above by calling a number of service providers. However, this service is also required to call an additional data collection service that stores the information for later retrieval but accepts only individual customer records. We can achieve this scenario using the Fan-Out primitive (shown in Figure 4-3).

A copy of the message is routed to the Fan-Out primitive that is configured to fire the out terminal for each customer record. In this example, when the Fan-Out primitive receives the message, the Fan-Out primitive fires its out terminal four times. Each time, it places the customer record in a well known place in the flow message, called the Fan-Out context, so that it can be used to create the service request message before the ServiceInvoke primitive invokes the data collection service.

You can also combine the Fan-Out primitive with a Fan-In primitive (shown in Figure 4-4 on page 40) to aggregate messages from multiple sources. You can use the Fan-In primitive only with a Fan-Out primitive because this pairing defines the scope in which aggregation can occur. The Fan-In primitive can act as one of three types of decision points for when to
continue the process flow. To illustrate this function, consider the following scenario in which a service is required to enrich a holiday booking record with additional customer and location information before propagating this information to a holiday reservation service.

![Combining the Fan-Out and Fan-In primitives](image)

The Fan-Out primitive is configured to fire the out terminal just once. This terminal is linked to two flow paths. Each path creates the service request message before invoking the service and storing the response message in a well-known place in the message structure, called the *Shared context*, which is used specifically for the Fan-Out and Fan-In primitives.

You can configure the Fan-In primitive to fire its out terminal in different ways by specifying the following properties on the mediation primitive:

- **Count**
  Fires the output terminal when a set number of messages are received at the input terminal. However, the count behavior does not stop the Fan-Out primitive from sending messages. Therefore, it is possible that the Fan-In count decision point will be reached more than once, causing multiple firings of the output terminal. If the Fan-In primitive is configured to fire for two messages and one message is received from the customer and location services, then the out terminal is fired. However, if only one service returns a response message, then the Fan-In primitive fires its incomplete terminal.

- **XPath decision**
  Fires the output terminal if an XPath evaluation of the incoming message evaluates to true. For example, you can configure the Fan-In primitive to fire its terminal when records are returned from the location service at one or more locations.

- **Iterate**
  Waits to receive all messages produced by the corresponding Fan-Out primitive when in iterate mode. If the Fan-Out primitive is configured in iterate mode, then the Fan-In primitive waits until all messages are processed before firing its out terminal once.

The different types of decision points enable you to specify the precise behavior of the flow depending on your specific requirements.
4.2.4 Message transformation

By using the Business Object Map mediation primitive, you can transform messages from one type to another or augment an existing message structure. In Figure 4-5, the business object map editor shows the incoming message type being transformed into the required message type for the target service. The business object map editor provides a graphical representation of the mapping and enables you to define sub-maps so that you can reuse the mapping in other mediation flows.

![Business object map](image)

In addition, when you use the Business Object Map primitive, you can also access the relationship service, to define relationships between different representations of data that refer to the same entity. For example, you might have several systems that store and process order information. One system might represent the order ID as a numeric string. Another system might prefix the order number using three characters. Yet another system might use a six-digit character string, even through all three systems refer to the same order record. By using the relationship service, you can define the relationship between the different order ID representations and call this order ID from within a business map, which will map one form into another.

4.2.5 Refining message types

By using the SetMessageType primitive, you can refine message fields with type information. Type information is particularly useful when the message definition contains weakly typed field definitions but you know that the content of a particular data type will be in the instance message at run time.

Use of the SetMessageType primitive in a mediation flow is like performing a cast operation on part of the message, redefining a field from one data type to another. In the following scenario, a message contains a field called `input1` that is defined as an xsd:anyType element in its schema (shown in Figure 4-6 on page 42), allowing it to contain any data type at run time.
At run time, this field might contain two different types of record objects. Figure 4-7 shows a mediation flow that determines which type of record the field has by using metadata found in the incoming message. The message is routed to specific SetMessageType primitives that define whether the input1 element is a customer record or an order record and propagates this information to downstream mediation primitives.
In this example, the DefineCustomer SetMessageType defines the input1 field as containing a customer record (shown in Figure 4-8). This information is propagated to the downstream ModifyCustomer primitive so that the following data structure can be used to process the message. Notice how the input1 element contains a customer record object.

Similarly, the DefineOrder SetMessageType propagates the following schema-type information to the ModifyOrder primitive. In this example, the input1 element contains an order record object (shown in Figure 4-9).

By defining concrete types, you can specify the correct runtime message structure to be used by mediation primitives in the mediation flow.

### 4.2.6 Customizing capabilities

By using the Custom mediation primitive, you can add your own mediation function in a flow either by writing Java code or by using the visual snippet editor. The advantage of using the visual snippet editor is that you do not need to write code because the visual snippet editor contains predefined snippets that you can link to, in order to define the mediation behavior that you are trying to build.

You can define as many input and output terminals as you need to implement the required behavior. Figure 4-10 on page 44 shows the Custom primitive that is used to perform a common function on all fault messages returned from a service. In this example, the Custom mediation extracts the data from the incoming fault message that the service provider returns. The data is then placed into a standard-response message before being delivered back to the requestor.
Although this example uses one output terminal, you can define as many output terminals as you need and invoke them from the Custom primitive depending on a message condition that you define. For example, you can define an additional output terminal to propagate messages that fail because of custom validation logic in the primitive.

Many of the supplied mediation primitives enable you to promote configurable properties to the system administrator for modification either during installation time or at application run time. The Custom primitive also includes the ability to define properties and promote them in exactly the same way, allowing system administrators to alter the runtime behavior of the primitive based on the values of the promoted properties. In addition, custom mediation saves you time because you can copy and paste the primitive icon, which includes the user-defined code or visual snippets, into another mediation flow, instead of redefining the same logic.

4.2.7 Enhanced XSLT mapping editor

By using the XSLT mapping editor, which now has greater support for custom scripting, you can build sub-maps for reuse in other mediation flows. You can define conditional maps when you want to run a sub-map on an element in the message only if it meets a certain criteria. For example, you might want to alter the value of an order record but only on records that have an ID element of ABC. At run time, if the ID field meets the specified condition, the map is invoked and applied to the message contents.

Figure 4-11 on page 45 shows how you can define an inline mapping that moves the contents of the OrderRecord element from the input message into the output message. This map runs only when the message meets the condition that is specified on the map.
WebSphere ESB accelerates time to value by enabling you to create connectivity solutions between various service endpoints and applications and transform messages, all without writing code. WebSphere ESB also provides the tools you need to customize functions.

### 4.3 Expanding the reach of connectivity

This section discusses various communication protocols and message formats that WebSphere ESB supports. This extended support is accomplished by incorporating other IBM products and standard technologies into your runtime environment.

#### 4.3.1 Overview

To give the ESB the broadest scope, WebSphere ESB supports a wide range of communication protocols and message formats. It does this by integrating with WebSphere Transaction Extender, supporting third-party Java Message Service (JMS) providers, and enhancing HTTP support.
4.3.2 WebSphere Transformation Extender integration

Because WebSphere Transformation Extender, a powerful transformation and validation engine, is integrated with WebSphere ESB, you can now configure and use WebSphere Transformation Extender maps. These maps help you map external data formats to WebSphere ESB and WebSphere Process Server XML models with little or no custom coding in JMS, MQ, and file SCA data bindings.

4.3.3 Supporting third-party JMS providers

With the ability to interact with any JMS provider that supports the Application Server Facility specified by the JMS 1.1 standard, you can use your plug-ins for existing JMS bindings, namely data bindings and function selectors, without having to modify them.

4.3.4 Enhanced HTTP support

With HTTP bindings, you can easily connect systems that communicate by using HTTP. WebSphere ESB supports both the HTTP 1.0 and HTTP 1.1 standards and binary, XML, and SOAP payloads. In addition, you can secure communication using a Secured Socket Layer (SSL, and HTTP).

You can set HTTP headers either dynamically, by using the HTTP header in the service message object, or statically by configuring values on import or export. The mediation flow programming model also enables dynamic routing to different HTTP addresses by specifying the required endpoint address, which the import acts on, into the header of the service message object. As a result, messages can be routed to different endpoints based on the contents of the message header, which can be populated from various sources, such as a database, registry or message metadata.

As with the other protocol bindings, you must define how a received HTTP invocation communication maps to an invocation of an operation on a service interface. Supplied function selectors handle the common cases by examining the HTTP method (get, post, delete) and an HTTP header. When examining an HTTP header, the function selector reads the native operation from the TargetFunctionName HTTP custom header. This means that you can send messages to the same endpoint address but route them to different operations based on the value in the header.

4.4 Summary

With WebSphere ESB connectivity capabilities, you can use WebSphere Integration Developer to quickly and easily assemble service intermediaries that can connect service requesters and providers, even if the service requesters and providers do not share the same requirements for protocol, message format, or message structure. Furthermore, WebSphere ESB provides a scalable runtime environment where you can host, manage, and customize these intermediaries. This makes managing a modification or replacing a service straightforward so that your business can adapt to change more quickly.
Enabling connectivity with WebSphere DataPower

This chapter discusses how you can enable connectivity with WebSphere DataPower. WebSphere DataPower helps you perform the following tasks:

- Easily configure and manage your appliance to accelerate the return on service-oriented architecture (SOA) investments
- Improve DMZ deployments and security with a drop-in, hardened network device
- Achieve any-to-any data transformation between a wide range of data formats
- Implement multi-stage pipeline processing and content-based message routing
- Optimize bridging between wire-line messaging
- Facilitate robust SOA governance and quality-of-service capabilities
5.1 Introduction

Organizations have to respond quickly to tactical change, and they are adopting new operating models to achieve this agility. As a result, enterprises are recognizing the benefits of deploying reusable, open-standards-based software components in an SOA approach, from improvements to financial transactions and online shopping to cost-saving inventory optimization across suppliers and synchronized multi-channel product introductions. Likewise, embracing open standards, such as XML-based Web services, has helped many companies improve productivity, quickly respond to changing business needs, and seize opportunities as they arise.

To take advantage of the improved business processes, flexibility, and IT efficiency that come with moving to SOA, organizations require pervasive, scalable services and controls, robust security, and high service assurances in their infrastructures. Today, enterprises often find themselves struggling to deliver these critical SOA requirements while having to handle prohibitive cost, complexity, and hard-to-manage infrastructures. Addressing these challenges requires a pragmatic approach to SOA—one that simultaneously recognizes the evolution of standards, the value of existing infrastructure investments, organizational challenges, and how performance can be affected across applications.

WebSphere DataPower SOA Appliances redefine the boundaries of middleware by extending the IBM SOA foundation with specialized, consumable, dedicated SOA appliances that combine simplified integration, superior performance, and hardened security for SOA implementations. Designed to augment all phases of the SOA life cycle and implementation, these highly specialized devices combine a host of essential SOA functionality into a specialized appliance for easy consumption, deployment, and service delivery.

5.2 Facilitating manageable, secure, and scalable SOA solutions

This section provides an overview of WebSphere DataPower SOA Appliances and the WebSphere DataPower SOA Appliance portfolio:

- WebSphere DataPower Integration Appliance XI50
- WebSphere DataPower XML Security Gateway XS40

5.2.1 Overview

WebSphere DataPower SOA Appliances provide a complete, hardware platform for delivering highly manageable, more secure, and scalable SOA solutions. As specialized SOA hardware, WebSphere DataPower Appliances provide many core functions to SOA deployments in a single hardened device, such as service-level management, routing, data and policy transformations, policy enforcement, and access control. Additional advantages of a hardware appliance in the network layer include the ability to perform more security and structural checks without performance degradation.

WebSphere DataPower SOA Appliances also offer high levels of security assurance certification, which many enterprises including financial services and government agencies require. The combination of high-performance hardware acceleration, simplified deployment,
and ongoing appliance management means faster, more secure performance with a reduced need for SOA programming skills and faster time-to-market for SOA benefits.

WebSphere DataPower SOA Appliances are rack-mountable network devices. They are tamper-resistant and cannot be taken apart and deployed in other servers. Because of their versatility and ease of deployment, WebSphere DataPower SOA Appliances is a cornerstone of a future-proof infrastructure that appeals to various groups with stakes in successful SOA deployment. Such groups include enterprise architects, network operations, security operations, identity management, and Web services developers.

WebSphere DataPower SOA Appliances offer the following core functions:
- XML and SOAP firewall
- Centralized access control, policy enforcement, and service-level management
- Protocol-independent message brokering
- High-performance, multi-step, wire-speed message processing
- Advanced WS standard support
- Transport layer flexibility
- Scalable, any-to-any message transformation

The WebSphere DataPower SOA Appliance portfolio (shown in Figure 5-1) includes the following connectivity products:
- WebSphere DataPower XML Security Gateway XS40
- WebSphere DataPower Integration Appliance XI50

---

**Figure 5-1**  WebSphere DataPower SOA Appliance portfolio

### 5.2.2 WebSphere DataPower Integration Appliance XI50

As a core offering in the IBM ESB product portfolio, the XI50 is a hardware ESB for simplified deployment and hardened security, offering the ability to quickly transform data between various formats, including XML, existing, industry standards, and custom formats. Visual tools help you describe data formats, create mappings between different formats, and define message choreography. The XI50 can transform binary, flat-text, and other non-XML...
messages, making it an innovative solution for security-rich XML enablement, ESBs, and mainframe connectivity.

XI50 includes the following core capabilities, as well as the capabilities of the XS40:

- **Any-to-any transformation engine**
  Parses and transforms arbitrary binary, flat-text, and XML messages including EDI, COBOL Copybook, ISO 8583, CSV, ASN.1 and ebXML. This engine is integrated with WebSphere Transformation Extender Design Studio.

- **Transport bridging**
  Bridges request and response flows to and from a wide array of protocols, such as HTTP, HTTPS, MQ, Secure Sockets Layer (SSL), IMS Connect, and FTP.

- **Integrated message-level security**
  Includes mature message-level security and access control functionality. You can filter, validate, encrypt, and sign messages to secure your high-value applications. Supported technologies include WS-Security, WS-Trust, Security Assertion Markup Language (SAML), and Lightweight Directory Access Protocol (LDAP).

- **Lightweight message brokering**
  Includes sophisticated multi-step message routing, filtering, and processing with multiple synchronous and asynchronous transport protocols. In addition, the appliance has a detailed logging and audit trail, including non-repudiation support.

### 5.2.3 WebSphere DataPower XML Security Gateway XS40

Securing applications with distributed software requires complex custom configurations and constant code patches deep within the enterprise. Because the XS40 is a self-contained gateway that applications share, it unifies security at the network’s edge with interfaces familiar to security staff.

The XS40 offers the following key functions, which are also included in the XI50:

- **Centralized policy management and enforcement**
  With this function, you can centralize security functions in a single drop-in device that enhances security and reduces ongoing maintenance costs. You can configure simple firewall and Web services proxy functionality by using a Web user interface that can be running in minutes. Alternatively, by using the power of XSLT, the XS40 can create sophisticated security and routing rules.

Combining support for WS-Security, WS-SecurityPolicy, WS-ReliableMessaging, WS-Policy and integration with leading policy managers and service registries such as WebSphere Service Registry and Repository, the XS40 is a policy enforcement and processing engine for securing next-generation applications. Manageable locally or remotely, the XS40 supports Simple Network Management Protocol (SNMP), script-based configuration, and remote logging to integrate seamlessly with leading management software.

- **XML and SOAP firewall**
  This function filters traffic at wire speed based on information from layers 2 through 7 of the protocol stack, from field-level message content and SOAP envelopes to IP address, port or host name, payload size, or other metadata. You can predefine filters and automatically upload them to change security policies based on the time of day or other triggers.
"Field-level XML security

The XS40 selectively shares information through encryption and decryption as well as signing and verification of entire messages or of individual XML fields. These granular and conditional security policies can be based on nearly any user-defined filter including content, IP address, and host name.

XML Web services access control

The XS40 supports various access-control mechanisms including XACML, Security Assertion Markup Language (SAML), SSL, LDAP, RADIUS, and client or URL maps. The XS40 can control access rights by rejecting unsigned messages and verifying signatures in SAML assertions.

Service virtualization

XML Web services require companies to link partners to resources without leaking information about their location or configuration. With the combined power of URL rewriting, high-performance XSL transforms, and XML/SOAP routing, the XS40 can transparently map services to protected resources.

Data validation

With its unique ability to perform XML schema validation as well as message validation at wire speed, the XS40 ensures that incoming and outgoing XML documents are valid and properly structured, protecting your organization against threats, such as XML Denial of Service (XDoS) attacks, buffer overflows, or vulnerabilities created by deliberately or inadvertently malformed XML documents.

SSL acceleration

The XS40 scales transport layer security by accelerating bidirectional and mutual SSL transactions in hardware. You can configure the XS40 with multiple SSL identities functioning as client or server, with SSL policies based on message content or metadata, such as port number, and HTTP header.

5.3 Accelerating time to value

This section discusses the business value of WebSphere DataPower Appliances and how the products simplify your overall SOA infrastructure.

5.3.1 Overview

By integrating many of the core functions that organizations require to adopt SOA or Web services, WebSphere DataPower Appliances simplify the overall SOA infrastructure in a single device. WebSphere DataPower Appliances easily deploy into an existing environment as an inline network device. You gain business value from using these appliances without having to change your network or application software. As a result, proprietary schemas, coding, or application programming interfaces (APIs) are not required to install or manage the device.
5.3.2 Configuration-driven approach to SOA

WebSphere DataPower Appliances provide a robust mediation architecture with high ease of use. You declaratively define service mediations according to the notion of a flow of basic mediation actions. An action can correspond to basic functions of intermediary processing, such as routing, decryption, or logging. By using these blocks and the WebSphere DataPower Web user interface, you build collective flows, augmenting actions as processing steps. By taking a configuration-driven approach, with WebSphere DataPower appliances, you can quickly implement common SOA use cases with minimal programming required. Figure 5-2 shows the palette of common mediations (actions) that you can drop in the message processing policy.

![Figure 5-2 Palette of common mediations](image)

In addition to the Web user interface, WebSphere DataPower provides the following other administrative configuration interfaces. They address the divergent needs of different groups, such as enterprise architects, network operators, security operators, identity managers, and Web services developers:

- A command-line interface that you can access by using Secure Shell (SSH) and Telnet
- Programmatic support that is enabled through XML management interfaces, such as Service-Oriented Management API (SOMA) and Appliance Management Protocol (AMP)
- Multiple appliances that you can manage collectively by using IBM Tivoli Composite Appliance Management System Edition (ITCAMSE) for WebSphere DataPower

5.3.3 Enhanced standards support and interoperability

IBM recognizes that SOA must address the need that organizations have to integrate heterogeneous environments, both within and outside the enterprise. The WebSphere DataPower SOA Appliance portfolio has a long history of support for key and advanced standards including the following standards:

- WS-Security
- WS-Policy
- WS-ReliableMessaging
- SOAP
Additional third-party interoperability capabilities include Universal Description, Discovery, and Integration (UDDI) registries and databases, such as IBM DB2®, Oracle®, and Sybase.

5.3.4 Integration with development and deployment environments

Across the IBM SOA foundation, the XI50 has been integrated with popular development environments, such as the IBM Rational® portfolio and WebSphere Transformation Extender Design Studio. This integration reduces the time that you have to spend in development and debugging. In addition to interoperability, WebSphere DataPower SOA Appliances are also integrated with WebSphere MQ, WebSphere ESB, WebSphere Message Broker, and DB2 to help process SOA transactions in a faster, more secure, and simplified way. Additionally, these appliances enable you to take advantage of the IBM autonomic computing self-management capabilities, creating infrastructures that require minimal intervention, which lowers cost of ownership and improves service availability.

5.4 Expanding the reach of connectivity

This section provides details about WebSphere DataPower XI50.

5.4.1 Overview

WebSphere DataPower XI50 is a hardware ESB for simplified deployment and hardened security, which gives you the ability to quickly transform data between various formats including XML, existing, industry standards, and custom formats. The XI50 provides core ESB functionality including routing, bridging, transformation, and event handling. The XI50 provides a reliable, performance-oriented solution to many integration challenges. Because it is not limited to handling just XML, the XI50 helps IT organizations that need to benefit from the connectivity of SOA deployments but must also deal with their current reality of managing multiple proprietary, industry, company-specific and existing data formats. The XI50 is a drop-in integration point for such environments, reducing the time and cost of integrations and speeding the time-to-market for services.

5.4.2 Transforming messages

The XI50 transforms messages from one format to another (shown in Figure 5-3 on page 54), including binary and non-XML data, which provides the following benefits:

- Brokers data between previously discrete (silo) systems
- Simplifies the reuse of and connectivity to existing systems
- Promotes loose coupling of applications and services
- Transforms data on the wire, enabling integration without coding
You can use visual tools, such as WebSphere Transformation Extender Design Studio, to describe data formats, create mappings between different formats, and define message flows. Furthermore, with native connectivity to systems, such as IBM DB2 and System z™ (by using IMS Connect or WebSphere MQ), the X150 service-enables existing systems by providing SOAP and XML to existing format transformations, such as SOAP to COBOL Copybook.

5.4.3 Content-based routing

You can configure the X150 to perform content-based routing based on both protocol metadata and message content:

- Dynamically route messages that are based on context (for example, originating URL, protocol headers, and attributes) and message content (existing and XML):
  - XPath-based routing against part of the message content or context
  - XPath statements that point to dynamically set URLs and message queues (MQ, JMS)
  - One-way routing when a response from the service is not necessary
- Configure a routing table where routing parameters are supplied using XML. The use of a routing table turns routing changes, including transport protocol conversions, around quickly.
- Dynamically retrieve routing information from other systems, such as service registries, databases, Web servers, and file servers (shown in Figure 5-4).
5.4.4 Protocol bridging

As messages flow through the appliance, the XI50 takes a configuration-driven approach to protocol bridging multiple protocols and transports, such as HTTP, SOAP, MQ, JMS, FTP, and ODBC. It provides an intuitive, Web-based interface for configuring connectivity to the various protocols (shown in Figure 5-5).

Figure 5-5 The XI50 with a configuration-driven approach
Furthermore, the XI50 gives you the ability to quickly configure different protocols for any mediation flow. Figure 5-6 shows how you configure XI50 connectivity to WebSphere MQ.

Figure 5-6  Configuring XI50 connectivity to WebSphere MQ

Figure 5-7 shows you how can quickly configure an HTTP-to-MQ scenario by selecting HTTP as the front protocol and MQ as the back protocol.

Figure 5-7  Quickly configuring an HTTP-to-MQ scenario

With this point-and-click approach, you can bridge multiple protocols without programming and achieve faster time to value by easily implementing a common SOA pattern.
5.5 SOA management and governance

This section discusses the ways that WebSphere DataPower SOA Appliances can support your SOA management and governance efforts.

5.5.1 Policy-driven Web services management and SOA governance

By centralizing Web service management tasks and policy enforcement and decoupling them from applications, your SOA infrastructure increases in flexibility and scalability while offering you improved insight, visibility, and control. Moving functions, such as access control, Web service management, security, and policy enforcement, onto WebSphere DataPower SOA Appliances means that IT architects, operations, security personnel, and business personnel can decouple these functions from the core business applications, simplifying development, deployment, and manageability.

5.5.2 Web service security and access control

WebSphere DataPower SOA Appliances provide a comprehensive Authentication, Authorization, and Audit (AAA) framework for securing access to SOA services. The modular AAA framework (shown in Figure 5-8) enforces who can access what service at what time and provides support for standards and protocols, such as WS-Security and LDAP.

![Figure 5-8 The AAA framework of WebSphere DataPower SOA Appliances](image-url)
To configure access control policies and integration with existing security infrastructure, you select the tokens, the authentication method, and then the authorization method (shown in Figure 5-9).

![Figure 5-9 Configuring access control policies and integration](image)

### 5.5.3 Web services policy

The Web services policy (WS-Policy) specification is a general-purpose model and syntax that you can use to describe and communicate the policies of a Web service. The XI50 and XS40 provide a flexible WS-Policy framework. With this framework, you can use industry-standard policies for centralized SOA governance and management across services. In addition, IT can quickly access new and updated standard and custom policies for centralized management and enforcement by using WebSphere DataPower SOA Appliances.

The WS-Policy framework that XS40 and XI50 provide includes the following capabilities and benefits:

- Web-based user interface for configuring WS-Policy enforcement at the WSDL component level
- Support for WS-PolicyAttachment through embedded WS-Policy references, external attachment references, and policies stored in WebSphere Service Registry and Repository (WSRR) or UDDI registries
- Ready-to-use standard policy templates for WS-Security Policy and WS-ReliableMessaging Policy
Figure 5-10 illustrates how quickly you can configure the XS40 and XI50 to enforce WS-Policy for any given Web service. In this case, the requirement is to enforce a WS-Security Policy that requires the use of WS-SecureConversation when communicating with the StockQuote service.

**5.5.4 Web services interoperability compliance**

Web services interoperability (WS-I) is an open industry organization that promotes Web-service interoperability across platforms, vendors, and programming languages. To make their services more usable, enterprises are ensuring that their services conform to one or more of the WS-I profiles.

The XS40 and XI50 support WS-I Basic Profile and the WS-I Basic Security Profile, enforcing these interoperable service profiles and providing mediation capabilities to bridge compliant and non-compliant service providers and consumers. By providing centralized enforcement for these conformance policies, the XS40 and XI50 help you quickly ensure that your services are usable and interoperable without requiring changes to your Web service applications.
Figure 5-11 demonstrates how you configure the XS40 and XI50 to enforce WS-I conformance policies.

![Web Service Proxy Policy](image)

**Figure 5-11  Configuring the XS40 and XI50 to enforce WS-I conformance policies**

### 5.5.5 SOA registry and repository support

WebSphere DataPower SOA Appliances provide simplified integration with existing investments in SOA registries and repositories for retrieving service definitions, metadata, and policies for centralized policy enforcement. This integration includes support for WebSphere Service Registry and Repository and UDDI Version 3-compliant registries.

### 5.5.6 Web services management

As you move more and more services to production, you need to evaluate, manage, and enforce service-level goals across these services. WebSphere DataPower SOA Appliances provide an extensible, easy-to-use, service-level management (SLM) capability and standards-based Web services management framework that includes the following functionality:

- Graphical configuration at the WSDL component level
- Flexible actions when reaching a threshold: notify or alert, shape, and throttle
- Threshold for overall requests and failures
- Graphical display of captured metrics and thresholds
Figure 5-12 shows the Web Services Proxy SLM configuring the StockQuoteProxy request volume.

![Web Service Proxy SLM](image)

WebSphere DataPower Web services management framework also supports integration with management consoles, such as IBM Tivoli Composite Application Manager for SOA.

### 5.6 Summary

WebSphere DataPower SOA Appliances support enterprises as they embrace SOA, with a portfolio that can minimize the complexity of implementing SOA infrastructures. WebSphere DataPower SOA Appliances are based on open standards and can help lower the barriers for SOA deployments by simplifying and accelerating the underlying infrastructure that is required to process XML, Web services, and existing messages that form the basis of SOA.
Enabling connectivity with WebSphere Adapters

This chapter discusses how you can enable connectivity with WebSphere Adapters. WebSphere Adapters helps you perform the following tasks:

- Access information from various sources in a uniform manner
- Quickly discover services that are available on your enterprise information systems (EIS)
- Build applications that access technologies in just a few steps
- Automatically deploy applications to WebSphere Process Server and WebSphere Enterprise Service Bus (ESB)
- Use one adapter to access data sources from multiple IBM products
6.1 Introduction

To integrate business applications and processes, you must first connect your disparate application and information assets to a coherent framework. Then you can use these assets in new and flexible business service applications and extend and grow the return on investment in those solutions. In effective SOA solutions, new business service applications use WebSphere Adapters to integrate multiple enterprise business application suites, an ESB, business process server, or data transformation engines. WebSphere Adapters service-enable applications and are insulated from the details of information storage and retrieval.

WebSphere Adapters are software components that are based on the Java EE Connector Architecture (JCA) 1.5 specification. They provide client applications with bidirectional access to external services, transactionality, security, and high quality of service. The set of services can be available on either EISs from independent software vendors, such as Siebel, Oracle, PeopleSoft, SAP, and JD Edwards®, or through a set of technology APIs, such as e-mail, FTP, Flat File, or Java Database Connectivity (JDBC). With WebSphere Adapters, you can easily connect to these systems from the integration applications.

WebSphere Adapters provide streamlined service discovery and configuration that enables fast integration with external data sources. It enables WebSphere Message Broker Toolkit, WebSphere Integration Developer, Rational Application Developer, WebSphere Process Server, WebSphere Enterprise Service Bus, WebSphere Message Broker, WebSphere Application Server, and WebSphere Transformation Extender to use the same JCA adapter. This set of adapters supports most of your integration needs and can be extended, as needed, by building custom adapters with the WebSphere Adapter Toolkit.

6.2 Accelerating time to value

This section discusses the WebSphere Adapters portfolio, its value, and how these products present a consistent view of dissimilar systems.

6.2.1 Overview

WebSphere Adapters reduce the time and complexity to develop integration applications connecting to external services. Adapters abstract the view of dissimilar systems, their different interfaces, programming models and underlying technologies, and present the same, consistent view of these services to you. This view matches the programming model of the run time and can be easily integrated in the application.

The services that use adapters are created with the service discovery tools that are optimized for the usage patterns and recognize the differences between discovering services from the EIS metadata repository and building services based on the access to technologies. The
Chapter 6. Enabling connectivity with WebSphere Adapters

6.2.2 WebSphere Adapters portfolio

WebSphere Adapters provide access to major EIS systems and technologies in the following representative products.

- WebSphere Adapter for SAP Software
- WebSphere Adapter for Oracle E-Business
- WebSphere Adapter for Siebel Business Applications
- WebSphere Adapter for JD Edwards EnterpriseOne
- WebSphere Adapter for PeopleSoft Enterprise
- WebSphere Adapter for Email
- WebSphere Adapter for FTP
- WebSphere Adapter for Flat Files
- WebSphere Adapter for JDBC
- IBM CICS J2EE ECI Resource Adapter
- IBM IMS Connector for Java

6.2.3 SAP Software

WebSphere Adapter for SAP Software provides multiple ways to interact with applications and data on SAP servers. For outbound interactions, the adapter can issue remote function calls (RFCs) through the Business Application Programming Interface (BAPI®). The BAPI interface can work with individual BAPI, with ordered sets of BAPIs, or with the BAPI result set interface.

By using the Query interface for SAP Software, you can retrieve data from specific SAP application tables and data from an SAP table without using an RFC function or a BAPI. By using the Application Link Enabling (ALE) interface, your application can exchange data by using SAP Intermediate Data structures (IDocs). In addition, by using the Advanced event-processing interface, you can send data to the SAP server to be processed by an ABAP™ handler.

For extracting data from SAP and sending it to a service, WebSphere Adapter for SAP Software provides three interfaces:

- Synchronous callback interface
  With this interface, the adapter listens for events and receive notifications of RFC-enabled function calls.
- ALE inbound processing interface
  With this interface, the adapter listens for events and receives one or more IDocs.
- qRFC interface
  With this interface, the adapter can receive IDocs from a queue on the SAP server.

6.2.4 Oracle E-Business

WebSphere Adapter for Oracle E-Business Suite works directly with the database components in Oracle E-Business Suite by using WebSphere Adapter for JDBC.
The Oracle E-Business solution communicates with the Oracle E-Business Suite database by using JDBC. It shows you how to set up the interaction between the WebSphere Adapter for JDBC and the Oracle database using sample applications. The samples provide a guideline for database integration with the Oracle EBS modules and an Oracle API.

6.2.5 Siebel Business Applications

WebSphere Siebel Adapter uses Business Service, Integration Object, Business Object and Business Component interfaces provided by Siebel Business Application to provide bidirectional connectivity. This adapter uses Siebel Java Data Bean technology to interact with Siebel Object Manager and enable data exchange for the different interfaces.

A business component defines the structure, the behavior, and the information displayed by a particular subject, such as a product, contact, or account. Siebel business components are logical abstractions of one or more database tables.

Business objects are customizable, object-oriented building blocks of Siebel applications. Business objects define the relationships between different business component objects (business components) and contain semantic information about, for example, sales, marketing, and service-related entities.

With a business service, you can encapsulate business logic in a central location, abstracting the logic from the data that the business logic might act upon. A business service is much like an object in an object-oriented programming language. A business service has properties and methods and maintains a state. Methods take arguments that can be passed into the object programmatically or, in the case of Siebel eAI, declaratively by way of workflows.

6.2.6 JD Edwards EnterpriseOne

By using WebSphere Adapter for JD Edwards EnterpriseOne, your applications can send outbound requests to JD Edwards EnterpriseOne servers. The adapter processes requests by using one of two types of business objects:

- Business functions
  This is a business object container that can contain one or more business objects that can be processed as a single transaction.

- XML Lists
  An XML List is a single business objects that can query a table and return multiple records.

6.2.7 PeopleSoft

WebSphere Adapter for PeopleSoft Enterprise uses the Component interface to access elements of the PeopleSoft system. The interface exposes a component, which is a set of entities grouped for a business purpose, such as an employee profile, to the adapter and external applications. The adapter can access all business objects in the Component interface definition and PeopleCode methods that are associated with the underlying components.
6.2.8 E-mail

WebSphere Adapter for Email enables e-mail connectivity between WebSphere Process Server and one or more mail servers. The adapter can send e-mail to one or more e-mail addresses. You can also use the adapter to poll a mail server for incoming e-mail and then send the information in the e-mail to a service. The service uses the information that the adapter forwarded to complete a task.

The adapter sends e-mail by using the SMTP protocol. When polling a mail server, the adapter can use either the IMAP or POP3 protocol.

6.2.9 FTP

By using WebSphere Adapter for FTP, both WebSphere Process Server and WebSphere Enterprise Service Bus can access files that an FTP server manages without having to know the details of FTP communications or protocols. The FTP adapter provides support for Secure Sockets Layer (SSL) and Federal Information Processing Standard (FIPS) 140.

6.2.10 Flat Files

By using WebSphere Adapter for Flat Files, services running on WebSphere Process Server or WebSphere Enterprise Service Bus can exchange data with the local file system in two ways:

- Services can use the adapter to perform operations on files in the local file system, for example, to create, delete, or update a document or file.
- The adapter can poll a file system folder for new documents and send those documents to a service.

6.2.11 JDBC

By using WebSphere Adapter for JDBC, your application can access or modify data in a database. The adapter converts a request from the application to an outbound operation, which it runs to create, retrieve, update, or delete data in the database or to run a database program stored in the database. Processing these requests creates, retrieves, updates, or deletes rows in the corresponding database tables. The adapter also enables you to run stored procedures or store functions that are defined in the database, and to run user-defined SELECT, INSERT, UPDATE, and DELETE statements.

The adapter can also poll a particular table, called an event table, which contains changes to the data that need to be published. The event table can be a new table that is populated by the database application, or the user can define custom queries to treat an existing table as an event table.

6.2.12 CICS J2EE ECI resource adapter

By using the CICS ECI resource adapter, you can build and test CICS ECI applications that facilitate communication between J2EE applications and CICS transactions on servers. The Common Client Interface (CCI) can be used to communicate with any EIS. The CICS Transaction Gateway provides additional resource adapters that implement the CCI for interactions with CICS.
6.2.13 IMS Connector for Java

Java applications running on WebSphere Application Server use the IMS resource adapter (also called the IMS Connector for Java) to access or run IMS transactions that are hosted on IMS systems. The resource adapter provides global transaction and two-phase-commit support. It also supports component-managed and container-managed security.

6.2.14 Adapter delivery in WebSphere Integration Developer

WebSphere Adapters are delivered fully integrated with WebSphere Integration Developer. When you start the external service wizard, the adapters are automatically imported into workspace. You can also decide which adapter to use by selecting from the palette in the Assembly Diagram (shown in Figure 6-1).

![Figure 6-1 Selecting adapters from the palette in the assembly diagram](image)

6.2.15 Adapter tools

With the external service wizard, you can build applications and access external data sources through the adapter. The wizard is accessible directly from the assembly editor palette, as shown in Figure 6-1, or from the menu. Simply dragging the appropriate adapter element invokes the external service wizard.

The wizard flow depends on whether the EIS system provides a repository of metadata that you can explore to discover services (for example, SAP or JDBC) or whether the services need to be built from available data definitions (for example, FTP or E-mail).
6.2.16 Discovering services

The external service wizard (shown in Figure 6-2) helps you discover the metadata on the EIS. It prompts you only for information that is absolutely necessary (for example, the server location or user name) and uses the default values otherwise. With fewer questions to answer, you can build services rapidly. If necessary, you can fully configure the discovery process and created services.

![Figure 6-2 Specifying the configuration properties for ORDERS05](image)

To further help with the service development, the properties and settings you asked for are explained directly in the wizard. In addition, the WebSphere Adapters documentation is integrated with the WebSphere Integration Developer documentation and is available as context-sensitive help. Now you can look up more information about a property while you complete the wizard (shown in Figure 6-3 on page 70).
The flow of the external service wizard is specific to the EIS and the selected interface. However, the flow always follows the same structure, making your experience consistent:

1. Configure connection characteristics to access the EIS system.
2. Select objects or functions to be exposed as services.
3. Configure them and specify the import characteristics.
4. Select deployment and service saving options.

### 6.2.17 Building services

How you create services with technology adapters, such as Flat File or FTP, differs from how you create services with application adapters. With technology adapters, there is no metadata repository that describes the available services. As the application developer, you compose, or build, services from the available data definitions, creating relationships between the data, which are service operations. You also specify the data transformation that needs to occur. You can share the following artifacts and reuse them between applications and adapters:

- Data handlers, which transform content, for example, converting between XML text and data object. The XML can be in the file or an attachment to the e-mail. You can reuse the data handler between different adapters.

- Data bindings, which are configurable and reusable entities that convert between a data object and a record that an adapter needs. In this process, data bindings can invoke data handler to transform the contents. Also, you can configure a specific data binding to call a specific data handler, for example XML. Then save the data binding and reuse it in several adapter instances.

- Function selectors, which are configurable entities that determine which operation should be invoked for the given event. For the WebSphere Adapter for Flat Files and WebSphere Adapter for FTP, you can use the FileNameFunctionSelector to resolve functions and names based on rules that match a filename pattern and the
EmbeddedNameFunctionSelector to call a data handler to perform the functions and name resolution.

The configuration for each of these artifacts is stored as an editable binding resource configuration in the tool, which makes adjusting to application needs extremely easy. For example, to change how an object is transformed only requires modifying the configuration of the data handler or modifying the data binding configuration to use a different data handler (shown in Figure 6-4). Reuse is supported because the configuration can be shared across multiple adapter instances.

Figure 6-4  Selecting a binding resource configuration type

With WebSphere Adapters, you can access several types of EIS applications and integration technologies. The adapters recognize the differences between application and technology adapters and customize how applications are built for both groups. When accessing an EIS, you can build services based on the capabilities of the EIS that the adapter discovers. When using a technology adapter, you can build a service that provides a bridge to that technology by using schemas that you define or import.
6.3 Easier service creation

This section discusses the key capabilities of WebSphere Adapters.

6.3.1 Overview

From creating services by using adapters and extending those services to deployment, WebSphere Adapters are easy to use. For example, the service creation wizard simplifies your tasks, streamlining them for rapid service creation and hiding, in most cases, the complexity of the native EIS interfaces. Available help and inline explanations guide you while you select and configure objects to build the service. When the application is created, the tools help you deploy it without your intervention.

6.3.2 Discovery setup

Application adapters rely on the libraries that the specific EIS vendors provide for connectivity. The external service wizard helps you easily configure these dependencies and automatically makes them available to the resource adapter (shown in Figure 6-5).

![External Service](image)

In most cases, you are given the name of the required library or you can browse for it on the file system. When you locate the required library, it is automatically configured for the resource adapter to use it.

6.3.3 Template-driven development

When building services based on the Flat File, FTP, and E-mail adapters, the wizard simplification is taken one step further. By taking advantage of the fact that these adapters are used most commonly in certain patterns, the tool provides the pattern wizard, which generates the service using minimal information from you (shown in Figure 6-6 on page 73).
Chapter 6. Enabling connectivity with WebSphere Adapters

Figure 6-6  Pattern wizard

The pattern wizard supports the following activities:

- Write to a local file
- Read from a local file
- Write to a remote (FTP) file
- Read from a remote (FTP) file
- Send an alert e-mail

For building services that are not included in the available pattern, you use the external service wizard.

For example, to write a business object as XML to file, you need to complete the following steps:

1. Select the pattern to use (shown in Figure 6-7), in this case, Create an outbound Flat File service to write to a local file.

Figure 6-7  Creating an outbound Flat File service

2. Select the service name and where the artifacts will be generated (the module name).
3. Specify the business object and where the resulting file should be written (shown in Figure 6-8).

![Image](image.png)

**Figure 6-8 Identifying business object and output directory**

4. Specify where to put the file, accepting the defaults for the remaining two panels, and then click Finish.

The process involves navigating only three wizard panels and completing only three fields, allowing for quick setup and requiring minimal technical skills.

### 6.3.4 Automatic resource creation

When the application containing the services that use the WebSphere Adapter is deployed to the server, WebSphere Process Server or WebSphere Enterprise Service Bus, the application can run immediately. Unlike traditional enterprise applications containing JCA Resource Adapters, which require you to configure various resources on the server, such as connection objects, all of the resources are created automatically and the application can run immediately when the WebSphere Adapters are deployed.

### 6.3.5 Shared adapters

With WebSphere Adapters, you can deploy adapters to the application server in the stand-alone mode, thereby simplifying maintenance. A single instance of the adapter is installed at the server and is available and shared by all applications that connect to the specific target system on that server.

### 6.3.6 Problem determination and monitoring

After adapters are deployed to the application server, you can monitor their performance using built in Application Response Measurements (ARM) support and performance metrics. The adapters can also emit events in Common Event Infrastructure (CEI). In case of problems, enhanced problem determination and monitoring support is provided with First Failure Data Capture (FFDC), along with extensive logging and tracing.
6.4 Extending the reach of adapters

This section discusses the key IBM products that work with some or all of the WebSphere Adapters:

- WebSphere Message Broker
- WebSphere Application Server and Rational Application Developer
- WebSphere Transformation Extender

6.4.1 Overview

WebSphere Adapters provide first-class integration with numerous EIS and technologies. With the tools and run time, you can integrate with these data sources in multiple IBM products that take advantage of adapter capabilities, versatility, and flexibility to fulfill their connectivity needs. The following IBM products, tools, and runtime environments work with some or all of the WebSphere Adapters, in addition to WebSphere Integration Developer, WebSphere Process Server, and WebSphere Enterprise Service Bus:

- WebSphere Message Broker
- WebSphere Application Server
- WebSphere Transformation Extender

6.4.2 WebSphere Message Broker

The WebSphere Adapters for SAP Software, Siebel Business Applications, and PeopleSoft Enterprise run and are delivered with WebSphere Message Broker. By using WebSphere Message Broker, you can discover the metadata that is available on the supported EIS and generate the connection definitions and the component that access the EIS. Then, you can create message flows that use those components, all without writing code.

The adapters are presented to WebSphere Message Broker as a set of input and request nodes (shown in Figure 6-9).

![Figure 6-9  WebSphere Adapters from Message Broker](image-url)
6.4.3 WebSphere Application Server

With Rational Application Developer connector tools, you can create applications by using WebSphere Adapters for SAP Software, Siebel Business Applications, PeopleSoft Enterprise, JD Edwards EnterpriseOne, and Oracle E-Business to run on WebSphere Application Server. The service discovery flow is the same as in the WebSphere tools, differing only in the launch point and the type of artifacts generated. The artifacts match the tool and runtime programming model, representing the data exchanged between the client application and the adapter.

6.4.4 WebSphere Transformation Extender

WebSphere Transformation Extender is the primary data transformation and validation engine for the WebSphere Portfolio. It uses WebSphere Adapters for Email, FTP, SAP Software, Siebel Business Applications, and PeopleSoft Enterprise on input and output cards, as a source or a target of the data transformation.

6.4.5 Building custom adapters

WebSphere Adapter Toolkit jump starts the development of JCA Resource Adapters in cases where a custom adapter needs to be created. Adapters that are developed with WebSphere Adapter Toolkit contain pre-coded functionality based on the IBM Adapter Foundation Classes (AFC). AFC is a set of class extensions that is built on top of the J2EE JCA specification and implements base JCA functionality that is common to all adapters, enabling adapters to use various services, such as reliable inbound event delivery or logging and tracing. When you build an adapter based on AFC, you can concentrate on interactions with the target system because the foundation classes handle the infrastructure requirements of the adapters. In addition, adapters that are developed with WebSphere Adapter Toolkit use the WebSphere Integration Developer tool environment, for a tool experience that is similar to that used for IBM WebSphere adapters.
In WebSphere Adapter Toolkit, which is a set of WebSphere Integration Developer plug-ins, you use the adapter development wizard to generate the adapter implementation skeleton and its deployment descriptor (shown in Figure 6-10).

![New Connector Project](image)

*Figure 6-10  The adapter development wizard in the WebSphere Adapter Toolkit*

By using the visual editor in WebSphere Adapter Toolkit, you can modify the deployment descriptor and the sample adapter with source code.

### 6.5 Summary

By using WebSphere Adapters, you can quickly make EISs service enabled by connecting them to WebSphere products as part of an SOA. The client applications of the adapter are isolated from all aspects of connectivity and how the transaction is enacted with the EISs. As a result, you can expediently set up and modify your integration solutions without extensive coding, obtaining maximum business flexibility.
Enabling universal data transformation with WebSphere Transformation Extender

This chapter discusses how you can enable universal data transformation with WebSphere Transformation Extender. WebSphere Transformation Extender helps you perform the following tasks:

- Accelerate development with code-free design and deployment
- Perform high-throughput processing of complex transforms and enhancements
- Describe all data types with an ontological data model
- Transform and validate data in the same process
- Deploy to multiple environments with ease
7.1 Introduction

WebSphere Transformation Extender is a universal data transformation and validation engine that delivers flexibility for your IT systems, resulting in business agility. Through a codeless, graphical approach to development, it handles high volumes of large, multipart documents with complex formatting-helping address the challenge of integrating your enterprise business systems.

WebSphere Transformation Extender minimizes the impact that integration imposes on an application. Developers no longer need to modify interfaces to deal with new message structures and file formats, nor do they need to write and test new preprocessing and post processing programs to validate the syntax and semantics of content exchanged between applications.

Business is better served when electronic documents that are exchanged between organizations can be validated before they are sent or received, eliminating the costly errors and corrective actions required to resend the documents. Validation checking is also important for businesses that must comply with regulatory and industry requirements for business-to-business transactions.

7.1.1 Process complex data without manual coding

WebSphere Transformation Extender excels at processing old and new formats that contain complex and detailed business information, including XML, non-XML, and mixed formats. Without a dedicated solution for transformation, you need significant programming skills to transform complex formats, such as a Health Insurance Portability and Accountability Act (HIPAA)-electronic data interchange (EDI) document or an SAP IDoc, to other formats.

In this scenario, you need to write specialized, format-specific programs to perform the preprocessing and post-processing tasks of checking content quality. The tasks associated with manually transforming formats all need to be administered and maintained over time, incurring development and maintenance costs. Under these circumstances, your business might find it difficult to implement new services because multiple applications need to be maintained and retested.

Many customers use WebSphere Transformation Extender, which maintains each transformation mapping, including content validation and format variations, in a portable form, regardless of sources, target applications, platforms or adapter requirements, and without the user having to write code.

7.1.2 Service-oriented architecture

WebSphere Transformation Extender exposes transformations as services inside service-oriented architectures (SOA). It also complements extensible stylesheet language transformation (XSLT), mediation capabilities of WebSphere-based enterprise service bus (ESB) and BPM solutions, as well as enhances business-to-business trading hubs.
7.1.3 Mainframe scalability

WebSphere Transformation Extender is available on IBM AIX®, Microsoft Windows, Linux, and UNIX operating systems. More significantly, the WebSphere Transformation Extender engine is available on the IBM System z platform for embedding in COBOL applications, batch processing, and integrating with IBM DB2 software, as well as participating in IBM CICS or IBM IMS transactions. WebSphere Transformation Extender also supports Linux for System z technology.

7.2 Understanding the basics

This section discusses these key concepts:

- Type trees
- Maps
- Cards
- Systems

7.2.1 Overview

It is important to understand several concepts before proceeding: type trees, maps, cards, and systems.

Design tools provide the GUI-based integrated development environment on which you can develop, import, test, and tune the performance of transformations and metadata. Then, you can deploy them to the processing engine for system test and production implementation.

7.2.2 Type trees

A type tree is a graphical data dictionary that contains metadata definitions of the structure of the inputs and outputs that contain the source and target data for integration solutions. Type trees enable you to precisely define highly complex data, such as EDI, SWIFT, XML, and existing data structures.

Each data object in the input or output data is defined as a reusable object in a type tree that uses properties to describe attributes, such as length, justification, possible values, delimiters, and the order of the data objects that make up a complex data object. You use the Type Designer to define properties for text or binary data, different character sets, data structures, and semantic validation rules (shown in Figure 7-1 on page 82).
Type trees can come from one of three sources:

- Manually created by using the Type Designer
- Automatically created by a type tree importer or the Database Interface Designer, which generates trees based on known metadata, such as text files, database catalogs, SAP R/3® IDocs, or COBOL Copybooks
- Predefined as part of WebSphere Transformation Extender Packs that support particular standards, such as NACHA, X12, UN/EDIFACT, SWIFT or HIPAA

Type Designer is the data object modeling tool that is used to create and manage type trees that, among other capabilities, define properties for data structures, define containment of data, and create data validation rules.
7.2.3 Maps

A map embodies the complete definitions of data objects and the rules for their transformation and can implement a wide range of integration functions. Such functions range from simple transformation to sophisticated integration solutions that involve multiple, heterogeneous inputs and outputs, rule-based routing, and complex interface structures.

To formulate transformation and business rules, you use Map Designer as the modeling component in which you specify the rules for transforming and routing the data object definition that you created in the Type Designer. Map Designer is also where you analyze, compile, and test the maps that you develop (shown in Figure 7-2).

Figure 7-2  Map Designer
7.2.4 Cards

You can use any number of inputs or outputs in a single map in an any-to-any fashion. A map contains a card for each input and output of the transformation. For example, the following map (shown in Figure 7-3) includes one input cards and two output cards showing the transformation between an X12 purchase order and a batch of ACH credits.

Figure 7-3  A sample transformation map
The card (shown in Figure 7-4) specifies the structure of the data that it represents (type tree or XML schema), as well as information about the source or target of the data, such as the particular adapter to use (GET for source) to access the data and the transaction settings for the source or target data.

Figure 7-4  Edit Output Card
7.2.5 Systems

A system is a set of maps (or subsystems) that are combined to form a data integration process flow. You define systems in the Integration Flow Designer (shown in Figure 7-5).

Each system can have one of two processing modes, Launcher or Command Server, depending on how the system is deployed. If a system is developed for the Launcher (event server), the system includes definitions of events based upon which map is triggered to run. As mentioned previously, these events might be time events (for example, run every 10 minutes or every Monday at 2:43 a.m.). They might also be source events (for example, a message appearing on a queue or a row being updated in a database) or compound events made up of a combination of time and source events.

The set of events that initiates a map in a system is called a watch. Each event that contributes to the initiation of a map is called a trigger. When the resource (file, database, or message queue) that is an input to a map changes state (is created or modified), the input event or trigger causes the Launcher to initiate a map. You can also use adapters to trigger maps because adapters have a listener function to detect the appropriate external resource change. However, there might be multiple triggers that must collectively exist before a map starts. The Launcher manages the coordination of these events to ensure that the correct set of circumstances (such as time and source events) has occurred before a map starts.
7.3 Industry solution acceleration

This section discusses industry-specific WebSphere Transformation Extender Packs that are available to help you develop and maintain solutions based on industry-standard, information-exchange formats.

7.3.1 Overview

Optional industry-specific WebSphere Transformation Extender Packs help accelerate time to value as you develop and maintain solutions based on industry-standard, information-exchange formats that you need to transform into internal business-application formats. These Packs can also accelerate the process of separating batched transactions into individual transactions by using an internal business application. Packs might include predefined format templates, conversion maps, and validation support for important industry standards that often indicate where quality of service is mandated by external industry organizations or government regulatory bodies.

WebSphere Transformation Extender offers the following industry-specific Packs:

- **Health Care**
  - HIPAA EDI, Health Level Seven (HL7), and National Council for Prescription Drug Programs (NCPDP)

- **Financial Services**
  - National Automated Clearing House Association (NACHA), Single Euro Payments Area (SEPA), Financial eXchange™ Protocol (FIX), SWIFTNet FIN, and SWIFTNet Funds

- **Insurance**
  - ACORD

- **EDI**
  - For administration, commerce, and transport (EDIFACT), X12, EANCOM, Trading Data Communications Standard (TRADACOMS), and Odette

As standards evolve and are updated, IBM provides timely updates to the Pack contents, helping protect existing investments year after year. Pack contents are imported into the Design Studio tools to help ensure that your business can quickly assimilate new industry-led requirements.

7.3.2 WebSphere Transformation Extender Trading Manager

Electronic commerce data flows from external sources to internal destinations and from internal sources to external destinations, and possibly to other internal organizations. WebSphere Transformation Extender Trading Manager is a client/server product that helps you manage and process electronic commerce data. It also helps you control the business-to-business integration of partner relationships and message flow, whether the trading scenario is internal and external, external and external, and internal and internal.

By using WebSphere Transformation Extender Trading Manager, you can audit, control, monitor, and view the entire business-to-business integration environment across the extended enterprise with secure data exchange fully integrated with infrastructure systems.
WebSphere Transformation Extender Trading Manager consists of two components:

- **Partner Manager**
  A graphical application used to administer partner and trade link information, create a comprehensive database of an organization's electronic commerce information, and manage e-commerce activity. You use Partner Manager to perform the following tasks:
  - Define the trading relationships with EDI partners, including details such as the version of each document, and control numbering schemes, tracking, error handling, status, and the post office used.
  - Maintain the information for each trading partner, including details regarding addresses, contacts, and the individual departments or other groups in trading-partner organizations.
  - Provide audit and control information collected in the Trading Manager database that is used to monitor ongoing inbound and outbound e-commerce transaction activity.
  - Provide alerts when error conditions occur in e-commerce inbound or outbound processing.

- **Message Manager**
  A runtime system that manages the integration of partner messages using a predefined system of maps that process electronic commerce data. You use these maps to validate and route data. Message Manager supports two distinct data flows:
  - Inbound data flow
    Typically represents the receipt of data in EDI or proprietary format from an external trading partner
  - Outbound data flow
    Typically represents the retrieval of data from an internal application system in its native format for subsequent conversion and routing to an external trading partner.

The information from Partner Manager is shared with Message Manager for validating, tracking, and routing e-commerce data. Information about this e-commerce activity is then incorporated into the Partner Manager database for monitoring and reporting.

WebSphere Transformation Extender Trading Manager complements WebSphere Partner Gateway, which provides AS1 and AS2 channel support for EDI users transitioning from value added networks (VANs) to Internet EDI.

### 7.3.3 Extended connectivity

WebSphere Transformation Extender provides a range of technology connectors and adapters delivered as a single package. It also provides optional enterprise application packs with template formats and connectivity examples for leading enterprise-information suites, including PeopleSoft, SAP, SAP Exchange Infrastructure, and Siebel. The SAP XI Pack is unique because it supports the WebSphere Transformation Extender engine as an embedded component of the SAP Web server, which satisfies all of the SAP user's transformation and validation needs when integrating applications other than SAP into an environment that is primarily SAP.

Integrate and validate applications other than SAP into an environment that is primarily SAP
7.4 Flexible integration and packaging

This section discusses the WebSphere Transformation Extender offering.

7.4.1 Overview

WebSphere Transformation Extender is packaged in different editions to suit the deployment mode that best fits your requirement:

- As an embedded engine inside any COBOL, C, or Java application
- As a batch-file processing engine called from a command script or System z job control language (JCL)
- As an event-driven server, either stand-alone or as an ESB or BPM extender

7.4.2 Transforming data for application programming

You can use WebSphere Transformation Extender for application programming to deploy transformations in custom programming environments. C, Java, Enterprise JavaBeans™ (EJB™), COBOL and .Net application programming interfaces (APIs) are available to engage the transformation engine to process a pre-built transformation. Available for all System z environments and recommended for enhancing IBM WebSphere Application Server applications, WebSphere Transformation Extender for Application Programming can also be embedded within original equipment manufacturer (OEM) offerings and is the edition.

7.4.3 Transforming data for Command Server

You can use WebSphere Transformation Extender with Command Server to run transformations from a command line, shell script, JCL on System z, or timer, or extend an existing application and infrastructure with minimal intrusion, and for fixed-window batch processing. WebSphere Transformation Extender with Command Server for System z is recommended for CICS, IMS and DB2 environments.

7.4.4 Transforming data with Launcher

WebSphere Transformation Extender with Launcher provides a stand-alone event server to host the WebSphere Transformation Extender engine. It includes the command line and script processing capabilities found in WebSphere Transformation Extender with Command Server. With similar but less-extensive capabilities than WebSphere Message Broker, the Launcher activates transformations when triggered by an event, which might be a file creation, a message arriving on a WebSphere MQ queue, a database trigger, or a scheduler. WebSphere Transformation Extender with Launcher can handle complex events and can support time- and availability-based event synchronization on combinations of triggers on multiple input sources.

Trading Manager is an application of WebSphere Transformation Extender with Launcher that is deployed with EDI industry Packs (EDIFACT and HIPAA EDI) as a business-to-business hub.
7.4.5 Transforming data for integration servers

WebSphere Transformation Extender for integration servers provides installation and integration support when you are combining WebSphere Transformation Extender with WebSphere Message Broker, WebSphere Enterprise Service Bus, and WebSphere Process Server.

7.4.6 Transforming data for WebSphere Process Server

Before WebSphere Transformation Extender Version 8.2, bindings were custom-coded with the aid of the WebSphere Transformation Extender SDK or with software assets from IBM service teams. Now these interfaces can be generated from within WebSphere Integration Developer, the development studio for WebSphere Process Server and WebSphere Enterprise Service Bus. In addition, the latest Eclipse-based Type Designer and Map Designer simply plug into the Eclipse workbench to provide a single tool experience for the developer.

The WebSphere Transformation Extender transformation engine is a powerful enhancement that complements the native Java mediation and business-rule capabilities of ESB and BPM products. It helps you handle larger documents and messages with more complex formats that are not based on XML.

WebSphere Transformation Extender for Integration Services can also enhance solutions built with the WebSphere Business Services Fabric, which includes WebSphere Process Server and WebSphere Integration Developer. All of these products are available on System z, under the UNIX System Services shell and Linux on System z. WebSphere Transformation Extender Design Studio can be ordered separately and is required for developing transformation maps.

Consider an example where an outside application sends data that contains customer records in a COBOL format to WebSphere Process Server. You can use the WebSphere Transformation Extender data binding to convert the raw COBOL data to business objects, which you can then use with WebSphere Process Server.

First, you create a module using WebSphere Integration Developer. Next, you define a business object to hold the customer data. For example, Figure 7-6 shows a Customer business object that is comprised of a first name, a last name, and an address, which is comprised of a street and a city.

![Figure 7-6  A sample business object assembly diagram](image)
The COBOL data is defined by a copybook. Figure 7-7 shows CUSTOMER.cpy displayed in Notepad.

![Figure 7-7  COBOL data in Notepad](image)

Logically the data looks like the Customer business object, but physically it is comprised of a number of fixed-length, padded fields.

In the next step, you develop a WebSphere Transformation Extender map that can convert the native COBOL data to a business object (shown in Figure 7-8). The Map Generator wizard in WebSphere ESB and WebSphere Process Server partially automates this process.

![Figure 7-8  Map generator for WebSphere ESB and WebSphere Process Server](image)
Business objects are defined by XML schemas, which are stored in the project workspace. When the Map Generator wizard is instructed to use the Customer business object, the correct namespace is specified, an arbitrary name for the native data is chosen, and a skeleton map is generated (shown in Figure 7-9). The map has one predefined output card with a type tree that is actually the schema that represents the Customer business object. No input cards are defined yet.

Next, you create a type tree from the supplied COBOL copybook by using the WebSphere Transformation Extender COBOL Copybook Importer. When a type tree has been generated, you create an input card that uses the type tree, and then complete and compile the map.
Switching back to the WebSphere Integration Developer Business Integration perspective (shown in Figure 7-10), the next steps involve creating an interface and adding an operation.

![Business Integration perspective](image)

*Figure 7-10  Business Integration perspective*
After you create an interface and an operation (shown in Figure 7-11), you create the import and export for the module, which can share the newly defined interface. Assume that the COBOL data is expected to arrive through the WebSphere MQ queue, and that the XML you produce is destined for another queue. In this case, you must generate bindings for the import and the export.

![Figure 7-11 Creating an interface](image)

The export is expected to connect to WebSphere MQ, read raw COBOL data, invoke WebSphere Transformation Extender, and produce a business object. When choosing the data binding, note that the Map Generator wizard has already created a binding resource configuration.

Choosing the JMS Function Selector automatically and conveniently looks for an operation called handleText for any incoming message.

The import does not need to invoke WebSphere Transformation Extender. Therefore, the predefined "Business Object XML using JMSTextMessage or JMSBytesMessage" data binding suffices.

The module is ready to be built and deployed. You can run it by putting COBOL data onto the input queue. An XML representation of the Customer business object is sent to the output queue.
7.4.7 Transforming data for WebSphere Message Broker

With more extensive capabilities as an ESB, WebSphere Message Broker extended with WebSphere Transformation Extender helps you focus on strategic deployment across an organization. Complementing the Extended Structured Query Language (ESQL) and Java scripting capabilities of the compute node, the extended WebSphere Message Broker can process WebSphere Transformation Extender transformation maps from within a message flow as it runs in a WebSphere Message Broker run time.

The combination of this universal ESB and WebSphere Transformation Extender extensions provides the broadest capabilities for simple and complex routing and mediation. WebSphere Transformation Extender makes Industry Packs available to WebSphere Message Broker users, accelerating time to value for EDI and complex document-processing solutions. WebSphere Message Broker gives WebSphere Transformation Extender users a broad range of transports and routing capabilities, including secure publish-and-subscribe routing, WebSphere MQ, Java Message Service (JMS), and Web services. In addition, WebSphere Message Broker now provides simplified parsing and multiple input capabilities equivalent to those found in WebSphere Transformation Extender with Launcher.

Two components comprise the WebSphere Transformation Extender for WebSphere Message Broker:

- The Parser is analogous to the Data Handler (data binding) capability in WebSphere Transformation Extender for WebSphere Process Server because it can be bound to input and output nodes to convert data between native streams and message trees.
- The Node enables you to run WebSphere Transformation Extender maps with message flows.
Let us see how the WebSphere Transformation Extender Parser compares to the WebSphere Process Server Data Handler. We build the same example as you did to illustrate WebSphere Transformation Extender for WebSphere Process Server. We begin by creating a new message flow project and a new message flow (shown in Figure 7-12).

Figure 7-12  Creating a new message flow
We add an MQ Input node because you can assume that an external application will deposit COBOL data onto a queue. We can choose any other nodes to complete the flow. Therefore, we add a Compute node and an MQ Output node (shown in Figure 7-13).

The MQ Input node receives COBOL formatted data, but the Compute node expects a message tree. We can use the WebSphere Transformation Extender Parser to solve this format discrepancy, but first you need a type tree that understands the COBOL data.
As in the WebSphere Transformation Extender for WebSphere Process Server example, you can use WebSphere Transformation Extender COBOL Copybook Importer to generate a type tree that accurately describes the format of the copybook (shown in Figure 7-14).

![Figure 7-14 WebSphere Transformation Extender COBOL Copybook Importer](image)

The WebSphere Message Broker parsers expect a set, a type and, optionally, a format. The WebSphere Transformation Extender type tree is the set and the type tree is the type. The WebSphere Transformation Extender Parser does not require the format.

If the chosen type tree is opened in Type Designer, and a type is selected, an option has been added to generate a map for the WebSphere Message Broker parser. When you select this option, you are prompted about whether you want to create a map for WebSphere Message Broker deployment. Then a map is generated that is added to the current project. The wizard also compiles the map and creates a MAR file, which is also added to the current project.

Switching back to the Message Broker Application Development perspective, you can now complete the configuration of the input node (shown in Figure 7-15).

![Figure 7-15 Message Broker Application Development perspective](image)
The message domain is WTX: WebSphere Transformation Extender type tree. In addition, the message set is the name of the type tree you want to use, and the message type is the chosen type. We can disregard the message format.

After you configure the remaining nodes, you can build a broker archive for deployment (shown in Figure 7-16).

![Figure 7-16 Building the broker archive](image)

Notice that the BAR file editor prompts us not only for the message flow, but also for the MAR file. When the BAR file is deployed to the server, and the flow is started, the input node receives raw COBOL formatted data from WebSphere MQ, and then invokes the WebSphere Transformation Extender Parser, which runs the automatically generated map and produces a logical message tree representing the data that other nodes in the flow can use.

Within the WebSphere Message Broker environment, WebSphere Transformation Extender can also run as a node itself. WebSphere Transformation maps can accept input message trees or raw input messages if the WebSphere Message Broker BLOB domain is used and produces either output message trees or fully formed output messages (again, depending on which domain is used).
When working on a message flow in the WebSphere Message Broker Application Development perspective, a WebSphere Transformation Extender node is available on the palette (shown in Figure 7-17).

The WebSphere Transformation Extender node, like any other node, can be dragged onto the canvas and used in any message flow. It provides an alternative to the built-in Compute, JavaCompute, Mapping, and XSLT transformation nodes. With this node, you can re-host previously developed maps in the WebSphere Message Broker environment.
The message flow (shown in Figure 7-18) uses the WebSphere Transformation Extender node to run a map that transforms the data received from a WebSphere MQ Input node to another format that is required by the WebSphere MQ Output node.

Figure 7-18  WebSphere Transformation Extender map node
You can configure the WebSphere Transformation Extender node (shown in Figure 7-19) to use either a source map (from the Eclipse workspace) or a pre-compiled map (for compatibility with WebSphere Message Broker 6.0). If you choose a source map, it is compiled when the flow is compiled, and then it is added to a MAR file, which is added to the current project. Then you can add the MAR file to the Broker Archive and deploy it.

At run time, input nodes pass data to input cards. (You can use the Collector node to pass multiple inputs to multiple cards.) Input cards that are not wired to input nodes use their own native adapters to extract data when the map runs. Output cards pass data to other nodes, and output cards that are not wired use their own adapters to give data to external targets.

With the advanced options, you can cache maps, for example, so that they are loaded only when the Execution Group starts or a WebSphere Transformation Extender flow is started. You can do this for map settings (usually compiled into the maps themselves), such as trace and audit settings, to be overridden by WebSphere Message Broker. For example, if a message with a correlation ID is abc*xzy generates an output file called ABC*.txt and a message with a correlation ID of abc123xyz is received, WebSphere Transformation Extender produces an output file named ABC123.txt.

Logging information is sent to the WebSphere Message Broker logging mechanism (the Event Log in on Microsoft Windows or the system log on UNIX platforms).
7.4.8  WebSphere Transformation Extender Design Studio for DataPower

WebSphere Transformation Extender Design Studio can also be used to design maps for processing on WebSphere DataPower XI50 Appliances. WebSphere DataPower devices provide powerful security, XML acceleration, and ESB functionality in an easily maintainable environment. The appliances offer a wire-speed XSLT engine for XML-to-XML transformation. With WebSphere Transformation Extender Design Studio, you can now transform non-XML data in a graphical environment.

The Design Studio simplifies the transition from the WebSphere Transformation Extender runtime to WebSphere DataPower. You just have to modify one setting (map runtime), and you are ready to design for the WebSphere DataPower environment. By using the build map operation inside the Design Studio, you can verify that all of the mapping rules are supported by the WebSphere DataPower runtime. You can also run your maps on the appliance in this graphical environment. The run map operation sends data to the appliance, performs the transformation, and then displays the results all within the Design Studio tools. All of the mapping concepts that you are familiar within WebSphere Transformation Extender, such as type trees and maps, apply in this WebSphere DataPower, making the transition as seamless as possible.

7.5  Summary

WebSphere Transformation Extender solves difficult transformation problems quickly with one common design environment. It delivers consistent data transformation across the enterprise, independent of data structure, data location, and infrastructure. It enhances the WebSphere connectivity portfolio with advanced transformation and validation capabilities and provides the same capabilities for System z Batch, CICS, IMS, and DB2 environments. In addition, the industry Packs that WebSphere Transformation Extender provides help you deliver healthcare, financial services, insurance, and EDI solutions faster.
Enabling connectivity with WebSphere Service Registry and Repository

This chapter discusses how you can enable connectivity with WebSphere Service Registry and Repository. WebSphere Service Registry and Repository helps you perform the following tasks:

- Achieve business process flexibility and better change management early in the life cycle
- Automatically discover the dependencies among service artifacts
- Identify dependencies to perform and communicate impact analysis
- Mitigate business risk and gain control
- Extend the reach and visibility of service representations for applications connected to WebSphere MQ as services
- Discover services from WebSphere and .NET platforms for reuse and governance
8.1 Introduction

WebSphere Service Registry and Repository forms a core part of an organization’s implementation of a service-oriented architecture (SOA). It provides management and governance capabilities that enable you to get the most business value from your SOA. It facilitates storing, accessing, and managing service information, called service metadata, making it easy to select, invoke, govern and reuse your services.

Publish refers to the capability to publish services, related artifacts and the associated metadata. Find refers to the capability to find the services using the metadata. Enrich refers to the capability to enhance the run time and, as a result, business flexibility through efficient APIs for runtime lookup. Manage refers to the capability to manage services (and related entities, such as polices captured using WS-Policy documents) through impact analysis, change notification, version management, and the automatic update of runtime metadata (status and average response time) by integrating with the service monitoring products. Govern refers to the capability to control the service and related metadata throughout the service life cycle with fine-grained access control, the enforcement of metadata validation policies, and the automatic discovery of deployed services.

The publish and find capabilities of WebSphere Service Registry and Repository promote service reuse by providing greater visibility and easier access to existing services and form the central point of record for all of the services in your SOA. Runtime applications and processes have dynamic and efficient access to the service metadata that is around the service artifacts that are published into the registry.

You can use rich metadata capabilities to add classifications based on business objectives, user-defined properties, and relationships to other service artifacts, and exchanging this data with runtime monitoring tools and operational data stores. In doing so, you can optimize how you use services.

In addition to providing value to the connectivity portfolio of products, integrating with WebSphere Service Registry and Repository provides value to connectivity products such as WebSphere Enterprise Service Bus (ESB), WebSphere DataPower XI50 Appliance, and WebSphere Message Broker. The dynamic selection of services (for example, Web services), which is based on service information (or metadata), helps your systems make intelligent decisions when routing messages, thereby increasing the runtime flexibility of applications and processes (and MQ queues).

The ESB (in this case, either WebSphere Message Broker or WebSphere ESB) receives a message from a client application or a business process that requests a service that meets the criteria encoded in the message. The ESB sends the message to a mediation flow. Based on the request, the mediation flow looks up the candidate services in WebSphere Service Registry and Repository. For each of these services, WebSphere Service Registry and Repository provides information about the service in terms of its availability, usage criteria, and performance.

WebSphere Service Registry and Repository integration with WebSphere DataPower XI50 Appliance increases the runtime flexibility of integrated applications and processes and promotes reuse and eliminates redundancies by pushing internal and external Web and non-Web services.
enables policy enforcement and security for services stored in WebSphere Service Registry and Repository.

In addition, enterprise governance is enabled through key fine-grained capabilities, with which you can control the visibility of and access to services and their metadata. By using impact analysis, you can intuitively analyze the relationships between services to show the effects of additions, deletions, and modifications in service artifacts. The service life cycle, which you can customize to your business needs, enables you to govern services through all stages of construction, testing, approval, production, and retirement.

## 8.2 Accelerating time to value

This section provides details about WebSphere Service Registry and Repository and its capabilities, which you can use to speed your development activity.

### 8.2.1 Overview

Graphical visualization, quicker and faceted searching, simplified content creation and location using a home page, and enhanced document loading make WebSphere Service Registry and Repository even easier to use.

### 8.2.2 Graphical visualization

By viewing the content of WebSphere Service Registry and Repository graphically, you can simplify the visualization of relationships between artifacts and navigate the relationships between objects more easily. You can also achieve a unified view of all of the constituent parts of multiple-document Web Services Description Language (WSDL) and XML Schema Definition (XSD) files where entities like bindings and interfaces are stored in separate WSDL documents, for example.

In Figure 8-1, a WSDL document is selected as the root object. The graphical view shows both the other WSDL and XSD documents that the root object imports, as well as the logical objects that are derived from the root WSDL document.

![Graph for: ValidateInsurancePolicyProductionService.wsdl](image)

*Figure 8-1  WSDL document identified as a root object*
By hovering over an object, you can see basic information about that object (shown in Figure 8-2).

![Figure 8-2: ValidateInsurancePolicy basic information](image)

You can also see the names of relationships by hovering over the arrow that represents the relationship (shown in Figure 8-3).

![Figure 8-3: WSDL relationships to other objects](image)

If there are multiple relationships between two objects, you see a double-headed arrow. By hovering over one of these arrows, you can see all of the relationships between those two objects, in one direction (shown in Figure 8-4).

![Figure 8-4: Viewing relationships between two objects](image)
You can perform a number of actions on the currently selected objects in the graphical view (shown in Figure 8-5). Apart from refocusing the graph, all of the actions mimic the actions that are typically available on collection (tabular) views. You can select multiple objects by pressing the Ctrl key and then performing the following actions on a group of objects just as in the collection view:

- **Re-focus Graph** changes the current focus object in the graph and re-draws the graphical view starting at the selected object using the current options (only available for single selections).
- **Go to Details** displays the details view for the selected object (only available for single selections).
- **Add Properties, Add Relationships, Add classifications** take you to views or wizards that help you edit the metadata on the selected objects.
- **Add to Favorites** stores a reference to the currently selected objects in your favorites view.
- **Export** is to export the currently selected documents to the file system (not available for derived objects) to use service metadata in other applications.
- **Subscribe** is to subscribe to the currently selected objects to receive updates and notifications about changes to metadata.

![Figure 8-5 ValidateInsurancePolicy available actions and options](image)
You can navigate around more complicated graphs by using the viewing window, which provides an overview of the whole graph and outlines the section that is currently in view. You can change the view by clicking in the viewing window or dragging the outline to a new location (shown in Figure 8-6).

You can choose whether you want to display all objects, filter only the logical objects that are derived from documents, such as WSDL documents, or only document objects that are related to the selected document.

You can also choose the depth of the graph that you want displayed. The default option can draw up to 100 objects, which limits the amount of work that the client Web browser has to do to draw the graph. The Unlimited option shows all of objects that you can reach from the root.
object. The other options allow the depth to be fixed at 1, 2, 3, or 4 levels from the root object. In addition, you can set the orientation of the graph to either horizontal or vertical. Options are shown in Figure 8-7.

![Options for displaying objects](image)

**Figure 8-7 Options for displaying objects**

### 8.2.3 Quicker searching

You can now search for objects by name (shown in Figure 8-8) and retrieve a near real-time list of objects that match the search value. A similar auto-suggest feature is employed by popular Internet search engines. Customizable, the auto-suggest facility helps you find objects in the registry by name, without running a query.

![Searching for objects](image)

**Figure 8-8 Searching for objects**
8.2.4 Faceted searching

You can refine your searches by starting with a basic search that typically returns more objects than you want, and then iteratively refine the search by adding more criteria until you obtain the results that you want. You can save the final query and, subsequently, search with the saved search name.

Figure 8-9 shows the results of a basic search that returns numerous objects.

Each of the filters in the Filters section is relevant to at least one of the objects that is returned, with the exception of the generic Relationship and Property filters that are always present in the filter window. You can reduce the number of objects that match the query by clicking one of the filters, thereby adding the filter to the query criteria. The number of objects that match the filter is shown beside the filter. In the example, clicking the WSDL Document link reduces the result set to eight objects because there are only that many WSDL documents in the results list.

WebSphere Service Registry and Repository supports these filters:

- Classification
- Business Model
- Document Type
- Service Metadata
- Property
- Relationship

Figure 8-9  Repository search results
8.2.5 Creating and locating content

The home page is a configurable entity that helps you quickly create and locate content in WebSphere Service Registry and Repository from the start. It provides immediate access to the two most common actions that the registry provides, namely find and publish.

Figure 8-10 shows an example of a typical home page.

![Example of a typical home page](image)

The home page lists all of the classification systems that are available. By using these lists, users can quickly search by classification by clicking one of the top-level classes in a classification system. This selection takes you directly into a faceted search that is based on the results of that query with a single classification value.
Different sections of the home page show other types of queries that users can invoke. Clicking one of the values in the Business Objects section, for example, shows all of the instances of the selected business model template. The Saved Searches section shows the searches that have been saved with a specific name. The Service Documents section lists all of the supported document types. Clicking one of these document types shows you all of the instances of that type of document. Similarly, the Service Metadata section lists all of the supported logical object types, and clicking one of them shows you all of the instances of that logical object type.

In all cases, the results of the query are presented as a collection with a filter section so that the user can refine the query even further, if necessary.

You can customize all aspects of the home page on a per-role basis including what to show in the Welcome message or whether a particular page section should be present for a given role.

8.2.6 Load multiple documents

The load document wizard can load a .zip or .jar file that contains multiple documents at a time, which significantly reduces the time required to load groups of large documents. WebSphere Service Registry and Repository scans the documents in the .zip file and automatically detects their document types from the file extensions. To override defaults, you can provide a hint file inside the archive that explicitly declares the file types of the documents.

8.3 Easy administration

This section discusses various improvements to WebSphere Service Registry and Repository.

8.3.1 Overview

WebSphere Service Registry and Repository has greatly improved the ease of use for administrators when configuring the system by offering the following new facilities:

- New configuration perspective
- Classification system editor

8.3.2 Configuration flexibility

As the administrator, you can switch between different configuration settings in the registry and restore a good working set if a problem arises with the active configuration profile by configuring multiple profiles to be loaded into a single instance, with one of the profiles being the active profile (shown in Figure 8-11 on page 115). You can change the active profile by selecting and activating one of the other loaded profiles.
Figure 8-11  Configuration perspective

The first section in the configuration perspective navigation tree shows the currently active profile. In this section, you can update the individual configuration items.

In addition, you can use an editor to manage access control permissions, which shows administrators, at a glance, all of the roles that are defined in the system, all of the permissions that are defined for each role, and the users and groups that are members of each role. Plus, you can modify each setting using a Web interface rather than by relying on the WebSphere administrative command line.
8.3.3 Classification system editor

Use the classification system editor to create and update classification systems that have been loaded into WebSphere Service Registry and Repository. Figure 8-12 shows the initial display for a classification system in which you define the base URI of the classification system and enter labels and comments or descriptions in multiple languages.

![Figure 8-12 Initial display for a classification system](image)

When you select the Classes tab, you see the root classes of the classification system (shown in Figure 8-13).

![Figure 8-13 Root classes of the classification system](image)
You can use the following editing facilities for hierarchical ontologies without turning to an external tool:

- Add new root classes
- Add children to existing classes
- Change the parent of a class
- Delete classes

8.4 Improved service registration and integration

This section discusses WebSphere Service Registry and Repository integration capabilities.

8.4.1 Overview

Beyond registering and retrieving capabilities, WebSphere Service Registry and Repository can integrate with other registries because it includes the following capabilities:

- Support for WS-Policy
- Service discovery
- Synchronization with UDDI registries
- Enhancements to support for REST

8.4.2 Support for WS-Policy

Support for WS-Policy (Web services policy) now includes the following features:

- Extensions to the WSDL logical model to support all of the attachment points described in the WS-Policy and WS-PolicyAttachment specifications
- Support for syntax referring to WSDL elements that were introduced in WS-PolicyAttachment specification Versions 1.5

Figure 8-14 shows an example of a policy attachment that uses the WSDL element syntax.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsapolicy xmlns:wsap="http://schemas.xmlsoap.org/ws/2001/09/policy">
    <wsapolicyattachment extended":{"wsdl11" : "http://example.org/TicketAgent.wsdl11#wsdl11.portType(TicketAgent)"} />
  </wsapolicyattachment>
</wsapolicy>
```

Figure 8-14  Policy attachment that uses the WSDL element syntax

8.4.3 Service discovery

WebSphere Service Registry and Repository also improves time-to-value with support for service discovery, with which you can automatically publish metadata about services that are already deployed in the registry, showing you the services that are in use. You can discover Web services that are hosted by WebSphere Application Server or Internet Information
Services (IIS) and Service Component Architecture (SCA) modules that have been deployed to either WebSphere ESB or WebSphere Process Server.

In addition, you can run service discovery as either a one-off operation or on a regular basis, so that services deployed after the initial service discovery run are picked up the next time that the service discovery runs.

### 8.4.4 Performing more actions by using REST

You can retrieve documents by performing an HTTP GET of a URL that contains the bsrURI (the unique URI of the registry entity) of the document. You can use Representational State Transfer (REST) to perform the following actions:

- Create, retrieve, update, and delete the content (services and policies) and associated metadata
- Query the service metadata

You can request that the returned metadata is in either XML or JavaScript Object Notation (JSON) format. Figure 8-15 contains syntax that shows an example of the XML format for metadata.

```xml
<resources>
  <resource bsrURI="e8f4dec8-2656-4661.ba05.dd3b01dd057c" type="WSDLDocument">
    <properties>
      <property name="bsrURI" value="e8f4dec8-2656-4661.ba05.dd3b01dd057c"/>
      <property name="name" value="MyWebService.wsdl"/>
      <property name="namespace" value="http://test"/>
      <property name="description" value=""/>
      <property name="owner" value="UNAUTHENTICATED"/>
      <property name="lastModified" value="1109008849234"/>
      <property name="creationTimestamp" value="1109007953359"/>
      <property name="lastModifiedBy" value="UNAUTHENTICATED"/>
      <property name="encoding" value="UTF-8"/>
      <property name="location" value="MyWebService.wsdl"/>
      <property name="contentSize" value="2348"/>
    </properties>
  </resource>
</resources>
```

*Figure 8-15 XML format for metadata*
8.5 Service management and governance

This section discusses WebSphere Service Registry and Repository governance and service management capabilities.

8.5.1 Overview

WebSphere Service Registry and Repository includes capabilities for governance and service management, including the following functionality:

- Graphical view of impact analysis results
- Promotion
- User models

8.5.2 Graphical view of impact analysis results

In addition to seeing a graphical view of an object and its relationships, you can display the results of an impact analysis (shown in Figure 8-16) as a graph instead of as a table, achieving an intuitive way to visualize the relationships between the target object, its dependent objects, and the objects that depend on it in one integrated graphical view.
If you choose to include entities (WSDL, XSD, Interface) that depend on this entity or that this entity depends on, then the resulting graphical view (shown in Figure 8-17) shows the entity that is the subject of the impact analysis, in the context of dependent entities.

8.5.3 Promoting services

With the support for promotion in WebSphere Service Registry and Repository, you can automatically copy information from a central governance WebSphere Service Registry and Repository instance to different environment-specific instances. You can set up governance lifecycle transitions (for example, from Service is Planned to Service is Being Developed), so that they trigger the copying of information and automatically place the relevant information for the next stage of a service in the correct environment. The user benefits by being able to define metadata that is moved automatically between development, test and production environments.

For example, if an enterprise has three environments (development, test, and production) there are four instances in total. All of the metadata relating to a service across all of the environments accumulates in the governance instance and the appropriate subsets of the metadata are copied to each of the environment-specific instances as the service moves through its life cycle. If the metadata is not classified as relevant to a specific environment, then it is copied to all environment-specific instances. Metadata that is classified as relevant to a specific environment is copied only to the appropriate environment-specific instance.
Service endpoints are typically limited to one environment, so that each endpoint is classified with one endpoint and is copied to only the appropriate environment-specific instance. Figure 8-18 shows how services that have the appropriate metadata move between registries when their lifecycle state changes.

In addition, WebSphere Service Registry and Repository provides support for manually defined endpoints, so that you can define endpoints for services without having to update WSDL documents.

### 8.5.4 User models

In WebSphere Service Registry and Repository, user models, also called business models, are defined as Web Ontology Language (OWL) ontologies. OWLs permit strong typing of properties and relationships as well as cardinality constraints and default values so that you can represent any kind of metadata within WebSphere Service Registry and Repository.

By using user models, you can define custom object types that are specific to your business domain (for example, Customer or Department). You can also constrain the types of properties and relationships on the objects (for example, a date property must be of type date) and validate that the objects are correctly structured when they are created or updated.
8.6 Summary

WebSphere Service Registry and Repository provides management and governance capabilities that enable you to maximize the business value of your SOA. It facilitates storing, accessing, and managing service information so that you can easily select, invoke, govern, and reuse your services.

Based on a highly scalable and available architecture that is integrated on an IBM SOA foundation (an integrated, open-standards-based set of software, best practices, and patterns for SOA), WebSphere Service Registry and Repository provides essential registry and repository capabilities for your SOA.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

Online resources

The WebSphere software: Application Integration Web page at the following address is also relevant for further information:


How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads
ibm.com/support

IBM Global Services
ibm.com/services
Enable connectivity infrastructure for SOA solutions

Route and transform data and messages

Discover, manage, and govern services

Enterprise IT environments continue to grow in complexity, through natural expansion or mergers and acquisitions. As the pace of business demand accelerates, IT environments look more for opportunities to flexibly reuse existing applications, services, and data. With the IBM connectivity portfolio of products, these IT environments can discover and reuse services, expose and use application services, and route and transform messages. The set of capabilities (in the IBM connectivity portfolio) supports client requests for a broad service-oriented architecture (SOA) infrastructure that addresses the need to bridge and streamline communication in heterogeneous IT environments.

In this IBM Redpaper publication, we provide an overview of the IBM connectivity portfolio to market watchers who have a keen interest in understanding the most current connectivity technology releases, and how IBM is taking them to the next level. Specifically, we review the key benefits and features of the following products:

- IBM WebSphere MQ
- IBM WebSphere Message Broker
- IBM WebSphere Enterprise Service Bus
- IBM WebSphere DataPower
- IBM WebSphere Adapters
- IBM WebSphere Transformation Extender
- IBM WebSphere Service Registry and Repository

For more information: ibm.com/redbooks

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.