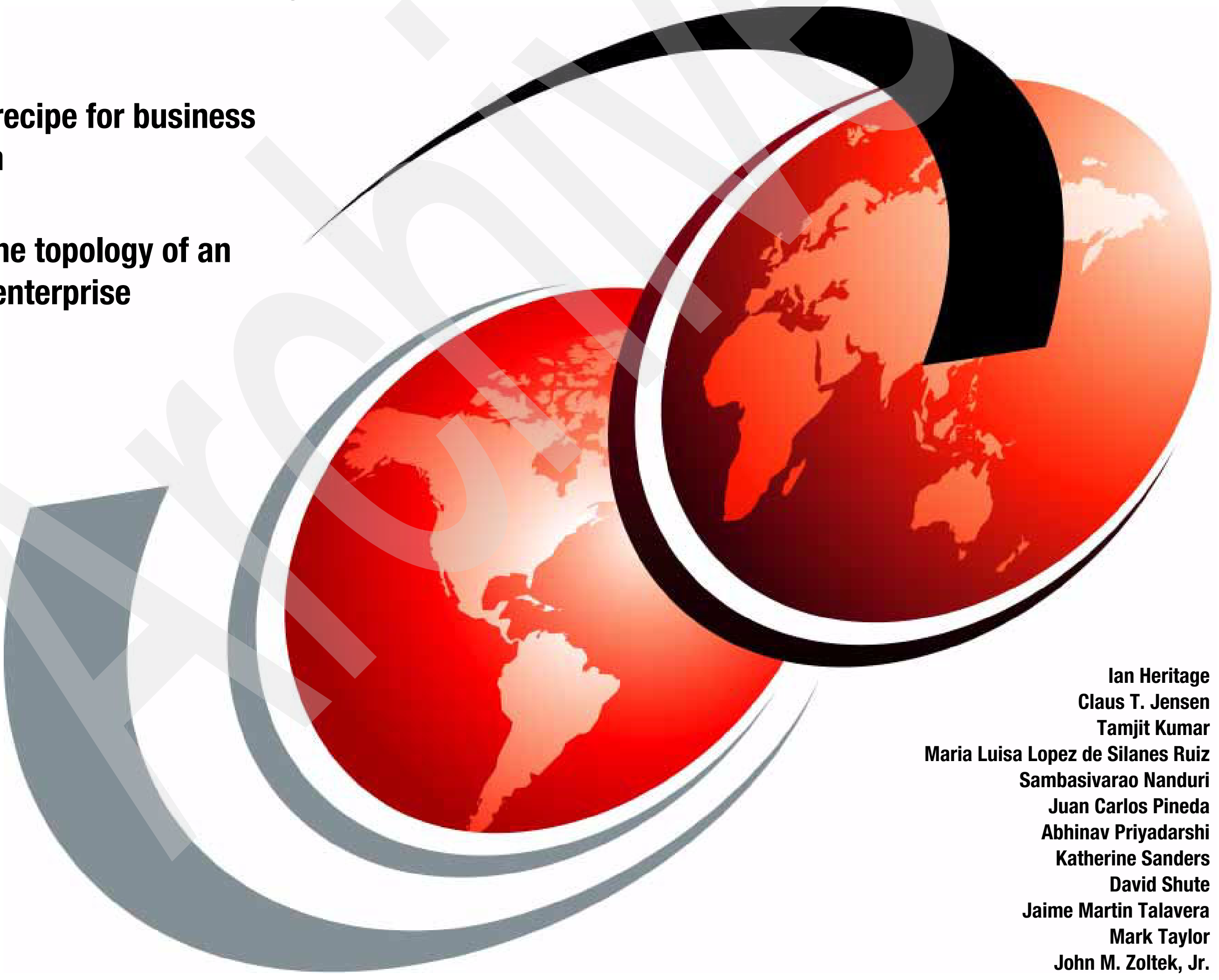
IBM

Integration Throughout and Beyond the Enterprise

Change the enterprise with integration patterns

Learn the recipe for business integration

Discover the topology of an engaging enterprise



Ian Heritage
Claus T. Jensen
Tamjit Kumar
Maria Luisa Lopez de Silanes Ruiz
Sambasivarao Nanduri
Juan Carlos Pineda
Abhinav Priyadarshi
Katherine Sanders
David Shute
Jaime Martin Talavera
Mark Taylor
John M. Zoltek, Jr.

Archived

Redbooks

ibm.com/redbooks

Archived

International Technical Support Organization

Integration Throughout and Beyond the Enterprise

April 2014

Archived

SG24-8188-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

Archived

First Edition (April 2014)

This edition applies to IBM Integration Bus Version 9, IBM WebSphere Registry and Repository Version 8, IBM WebSphere MQ Version 7.5, IBM WebSphere DataPower Integration Appliance XI52, and IBM MessageSight.

© Copyright International Business Machines Corporation 2014. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Archived

Archived

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

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

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

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.

Archived

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: 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®	Informix®	Redbooks (logo) ®
Cast Iron®	MQSeries®	System x®
CICS®	PowerHA®	System z®
ClearCase®	PureApplication®	Tivoli®
DataPower®	RACF®	WebSphere®
DB2®	Rational®	Worklight®
developerWorks®	Redbooks®	z/OS®
IBM®	Redpaper™	

The following terms are trademarks of other companies:

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Worklight is trademark or registered trademark of Worklight, an IBM Company.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Archived

Preface

Throughout the history of the IT industry, integration has been an important part of most projects. Whether it is integration of transactions, data, or processes, each has challenges and associated patterns and antipatterns. In an age of mobile devices, social networks, and cloud services, and big data analytics, integration is more important than ever, but the scope of the challenge for IT projects has changed.

Partner APIs, social networks, physical sensors and devices, all of these and more are important sources of capability or insight. It is no longer sufficient to integrate resources under control of the enterprise, because many important resources are in the ecosystem beyond enterprise boundaries.

With this as the basic tenet, we address these questions:

- What are the current integration patterns that help enterprises become and remain competitive?
- How do you choose when to use which pattern?
- What is the topology for a “composable business”?
- And how do you accelerate the process of implementation through intelligent choice of supporting integration middleware?

This IBM® Redbooks® publication guides integration practitioners and architects in choosing integration patterns and technologies. The book includes two parts:

- Part 1, “Topology of an integrated enterprise”, focuses on the enterprise ecosystem and the different types of integration tasks within and beyond the enterprise.
- Part 2, “Integration patterns”, defines the most common integration patterns and shows how to map these patterns to IBM integration technologies.

Archived

Authors

This book was produced by a team of specialists from around the world while working for the International Technical Support Organization, Raleigh Center.

Ian Heritage is an API management and SOA governance subject-matter expert who specializes in IBM API Management, IBM WebSphere® Service Registry and Repository, and IBM WebSphere DataPower®. Ian is based in Rochester, Minnesota, and works in the Worldwide WebSphere Technical Sales organization. He has helped customers in various industries plan and implement their API management and SOA governance strategies. Ian worked in IBM development for the first 10 years of his career, where he held several positions including, WebSphere Service Registry and Repository Integration Lead, Level 3 Service Lead, and Test Architect. He joined IBM in 2000 after graduating from the University of Hertfordshire, in the UK, with an MSC in Computer Science.

Claus T. Jensen is a Senior Technical Staff Member and Chief Architect for the SOA and WebSphere Foundation portfolio. The Foundation Architecture team has architectural responsibility for all IBM foundation software products to ensure that they are appropriately integrated and support the key design principles of SOA from a client perspective. Before joining IBM in March 2008, Claus was Group Chief Architect, VP of Architecture and Business Development, in Danske Bank, a regional European bank. He was responsible for guiding Danske Bank's SOA initiative and SOA Center of Excellence from its inception in 1999, and he is known as an SOA expert and evangelist. Claus is a member of the IBM Academy of Technology and holds a PhD in Computer Science from Aarhus University, Denmark.

Tamjit Kumar is a WebSphere consultant at Miracle Software Systems, Inc. in Novi, Michigan. Tamjit has more than eight years of experience in the IT field, including application design, development, and consulting. He holds a BE degree from the University of Delhi, in India. His areas of expertise are IBM WebSphere MQ, IBM Integration Bus, service-oriented architecture, and enterprise application integration. He has worked extensively on SAP integration with WebSphere Message Broker and migrating technologies such as Neon and DataGate to WebSphere Message Broker.

Archived

Maria Luisa Lopez de Silanes Ruiz has worked for IBM for 15 years, primarily as a lab-based consultant specializing in connectivity, integration, and business process management. Maria Luisa joined the IBM Integration Bus development team in 2012 to apply her client experience to improving the product documentation. Marisa has a Master's degree in physics, with a specialty in computer science, from La Universidad Complutense de Madrid.

Sambasivarao Nanduri is a certified IT specialist who has been working with IBM for more than 10 years. He started as an infrastructure administrator, overseeing the IBM AIX® system and network administration with SAN storage and IBM High Availability Cluster Multi-Processing for AIX clustering (now IBM PowerHA®), IBM WebSphere middleware stack, and IBM DB2® databases. Then he moved to the IBM Advanced Information Management (AIM) support team, where he focused on WebSphere Message Broker, WebSphere business events, and DataPower product support. He co-authored an IBM Redpaper™ publication about new of Message Broker features in Version 6.1, in 2009, and has participated in several WebSphere Technical Exchange webcasts and blogged on various topics, including security, performance, monitoring, and new Message Broker technologies, such as multi-instance brokers and eXtreme Scale.

Juan Carlos Pineda started working at IBM in February 2004 as a Java developer for the IBM System x® series inventory and management tools. He has more than 10 years of experience in the IT field, working with Java, Java 2 Enterprise Edition, and integration tools with IBM clients in Mexico and the US. In 2007, he became an integration developer with WebSphere Message Broker, WebSphere Transformation Extender, and WebSphere Partner Gateway. His articles about WebSphere MQ, MQ Link, and MQ Telemetry Transport have been published by IBM developerWorks®. He holds certifications as a Java developer and architect, as well as an IBM IT Architect, Experienced Level.

Abhinav Priyadarshi works for the IBM Integration Bus Development team at IBM India, in Bangalore. He has 13 years of experience with IBM. Abhinav develops new features, nodes, and hypervisors for IBM Integration Bus new releases. He also consults on WebSphere Message Broker and IBM Integration Bus EAI design and implementation for IBM clients in India and the Asia-Pacific region. He has worked in various client engagements and performed the implementation, using WebSphere Message Broker and related WebSphere products, including IBM WebSphere Transformation Extender and IBM Worklight®, in various integration solutions.

Archived

Katherine Sanders is a Level One Certified Senior IT Specialist based in the IBM Hursley Lab in the United Kingdom. She has been working as an IBM Software Services for WebSphere Consultant since May 2011. Before that, she spent seven years working in a variety of Software Engineering roles, including WebSphere MQ testing and WebSphere Application Server Development. She is currently specializing in IBM WebSphere Cast Iron® Cloud integration, IBM API Management, and WebSphere DataPower. She has published numerous articles through IBM developerWorks, IBM Redbooks and IBM Redpaper publications, and IBM blogs and has presented at a variety of conferences. She holds a Bachelor of Science degree in Computer Science from the University of Nottingham, in England.

David Shute is the Technical Enablement Program Manager for IBM WebSphere DataPower software. He has eight years of experience with DataPower and more than 20 years of experience producing and delivering educational materials about technology subjects. Mr. Shute has written several other IBM Redbooks publications, developerWorks articles, and numerous classroom and online courses. He has taught classes in SOA, XML, security, and application-level firewalls, worldwide.

Jaime Martin Talavera is part of the IBM Software Group's WebSphere Services team, based in Spain. He holds a Bachelor's degree in Computer Science from Comillas Pontifical University (ICAI) in Madrid, Spain. Jaime has more than five years of experience in the IT field, specializing mainly in integration technologies, such as IBM WebSphere DataPower, with IBM clients worldwide. He joined IBM in 2008 and has since written developerWorks articles and filed an IBM patent.

Mark Taylor has worked for IBM at the IBM Hursley laboratory in England for nearly 30 years, in various development and services roles. He wrote code for the early versions of IBM MQSeries®, porting it to numerous UNIX operating systems. He now works on product strategy and is responsible for defining functions that are included in new releases of WebSphere MQ and also develops some of the code for security features.

Archived

John M. Zoltek, Jr. is a Staff Software Engineer with the IBM Software Group in Research Triangle Park, in Raleigh, North Carolina, in the United States. He has 13 years of experience in Level 2 support and has worked on the Level 2 support teams for WebSphere MQ, WebSphere Embedded Messaging, the Service Integration Bus, WebSphere Service Registry and Repository, WebSphere Message Broker, and the IBM Integration Bus. He is active in IBM social media channels, writing blogs and tweets. He received a BS degree in Computer Science from Coastal Carolina University in South Carolina, in the US.

Additional contributors

Margaret Ticknor is an IBM Redbooks Project Leader for the International Technical Support Organization (ITSO) Raleigh Center, in North Carolina, US. She primarily leads projects about WebSphere products and IBM PureApplication® System. Before joining the ITSO, Margaret worked as an IT specialist in Endicott, New York. She attended the Computer Science program at the State University of New York at Binghamton.

Thanks to the following people for their contributions to this project:

- Bharat Bhushan
- Vivek Grover
- Fermin Luna
- Tanmayee Potluri
- Ozair Sheikh

Thanks to the following people for supporting this project:

- Judith Broadhurst, IBM Redbooks Editor
- Ella Buslovich, IBM Redbooks Graphics Editor
- Deana Coble, IBM Redbooks Technical Writer
- Richard Conway, IBM Redbooks IT Support
- Tamikia Lee, IBM Redbooks Residency Administrator
- Debbie Willmschen, Redbooks Process and Production Lead
- Redbooks Global Content Services Graphics Support Team, Bangalore, India

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Archived

Comments welcome

Your comments are important to us.

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

- Use the online **Contact us** form:

  ibm.com/redbooks

- Send your comments in an email message:

  redbooks@us.ibm.com

- Mail your comments:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS feeds:

  http://www.redbooks.ibm.com/rss.html

Archived

Part 1

Topology of an integrated enterprise

This part of the book focuses on the enterprise ecosystem and the different types of integration tasks within and beyond the enterprise. It covers the following topics:

Archived

Archived

1

Business integration in 2014

Young professionals check their bank balances while they are on trains, commuting to or from the cities where they work. The savvy shopper uses a price comparison service while in the supermarket. The child purchases movie-maker software for an iPad while visiting a friend. These scenes and many more have become everyday occurrences in modern society.

Terms such as *mashups* and *apps* have made their way into the English language. The volume of on-the-move information retrieval and other activities is growing exponentially. In 10 years, it might overtake the volume of more traditional electronic transactions. All of this tells you that modern technology changes the business model, not just the channel.

This chapter covers the following related topics:

- The confluence of forces
- Growing the ecosystem beyond the enterprise
- The API and Service Economy

Archived

1.1 The confluence of forces

What Gartner called the *nexus of forces* in the article cited by the following link is the confluence of mobile devices, social networks, cloud services, and big data analytics:

http://www.gartner.com/technology/research/nexus-of-forces/

This confluence implies an experience where context is key and important interactions might not be directly related to business transactions but rather focused on building social relationships and ecosystems. Furthermore, social interactions that are of interest to the business can and will happen between third-party connections. Just think about the potential impact, positive or negative, of a video gone viral.

Obviously, technologies such as mobile devices do change the channel, but more importantly mobile interactions happen in a real-time context. Consider the fact that you bring the device with you, always at hand. You can access any information or contact any friend, even if the interaction is asynchronous. Messages can be pushed, not just pulled, including traditional types of information, such as account balances. The growth of social media has created consumers who expect their opinions to matter and who choose mobile devices as the way of interacting with the world.

These characteristics change the ways that interactions need to be planned, managed, and monitored. More fundamentally, they change the scope of what is considered relevant to business success.

1.2 Growing the ecosystem beyond the enterprise

This confluence of forces several key questions and challenges for defining a business model and engineering the business solutions to support it:

- Q: Where do transactions happen?

  A: Anywhere and anytime. This is the essence of mobile and cloud technologies.
- Q: Who can influence your business?

  A: Anyone who publishes an opinion, positive or negative. That is the essence of social media.
- Q: Who can access your information?

  A: Anyone you can legally provide it to, whether in exchange for money, influence, or improved relationships. Applying big data analytics to vast amounts of available information sources lets you provide unique value and insight, particularly in the context of the Internet of Things, which is described in this video:

  http://www.ibm.com/smarterplanet/us/en/overview/article/iot_video.html
- Q: What is an *application*?

  A: Any piece of software that provides value (including mobile apps, software embedded in appliances or cars, cloud services, and so on).
- Q: Who is your developer?

  A: Anyone who builds business solutions by using your information or services. In an open ecosystem, it is often someone who is not employed within your own company.

Archived

Figure 1-1 Shows examples of these questions and challenges.

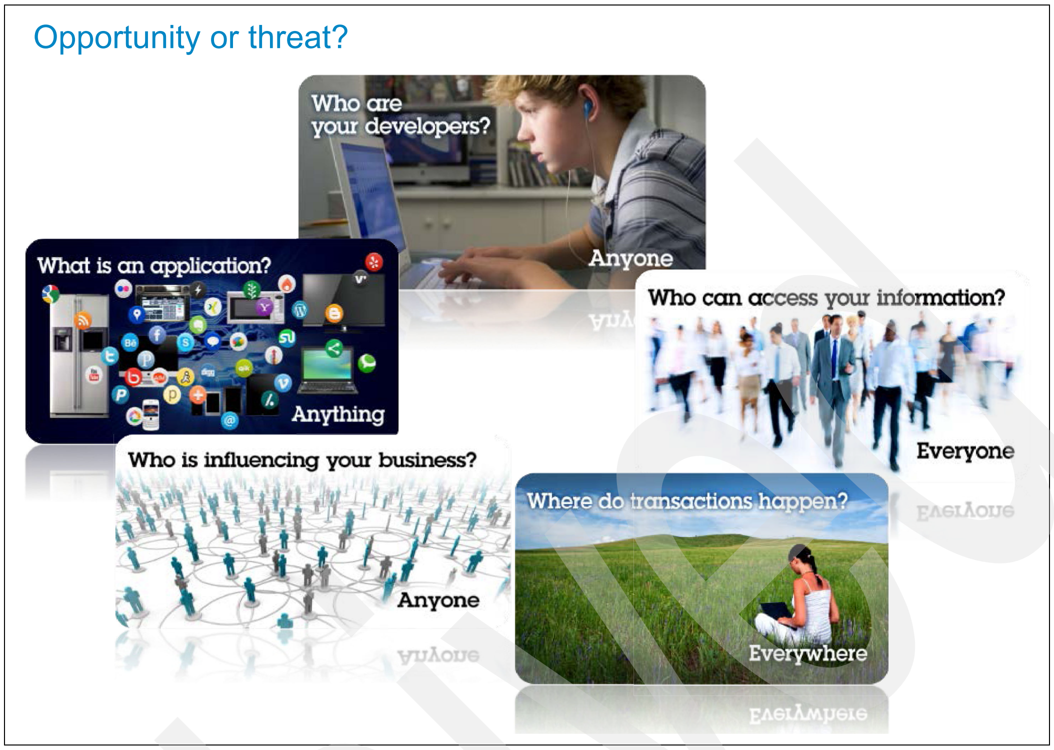


Figure 1-1 Opportunities for business transformation

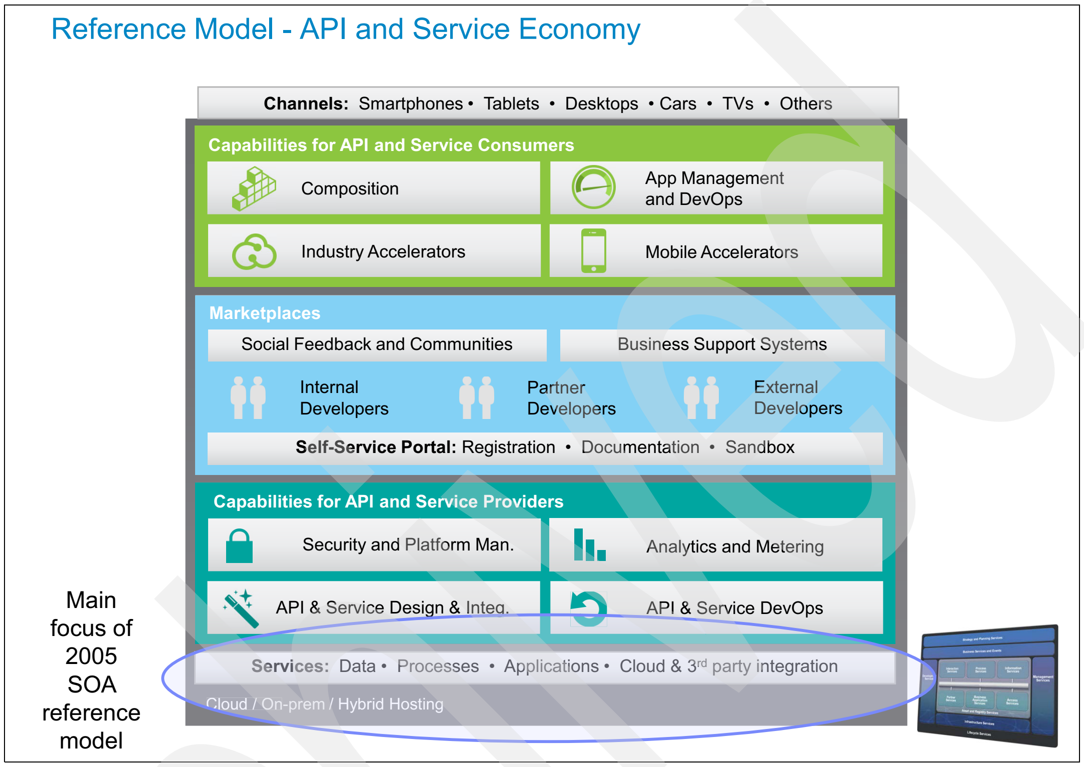
Taken together, the elements in Figure 1-1 represent a profound change in the way that we deliver solutions, as well as in what we deliver and what the business needs us to deliver.

A popular online store and a popular social networking service, although they use different business models, are both examples of early adopters of a computing model that is open by design and where the product is based on APIs and services that are projected into an extended ecosystem. Without its open merchant platform, the online store would not be the *one stop shop* for all kinds of goods and would probably not have become one of the dominant Internet retail portals. Without the open interface to its communication servers, the social networking service could not rely on various open source communities providing a myriad of smart clients, at no cost, and would probably not have achieved the popularity that it enjoys.

These are just two examples of a trend across all industries where solutions are, first and foremost, designed for an open ecosystem, whether those solutions are deployed internally, externally, or in a hybrid fashion. It is no coincidence that *mobile first* and *cloud first* are two of the mantras of this new age of computing. This age is marked by designing for a different experience and environment, yet allowing solutions to still run within the enterprise, where appropriate, or by using traditional channels.

1.3 The API and Service Economy

Embracing an open ecosystem, delivering APIs as part of a business product, and carefully promoting and managing your external business image all require more than integration middleware, as Figure 1-2 shows.



Reference Model - API and Service Economy
Channels: Smartphones • Tablets • Desktops • Cars • TVs • Others
Capabilities for API and Service Consumers
Composition
App Management and DevOps
Industry Accelerators
Mobile Accelerators
Marketplaces
Social Feedback and Communities
Business Support Systems
Internal Developers
Partner Developers
External Developers
Self-Service Portal: Registration • Documentation • Sandbox
Capabilities for API and Service Providers
Security and Platform Man.
Analytics and Metering
API & Service Design & Integ.
API & Service DevOps
Services: Data • Processes • Applications • Cloud & 3rd party integration
Cloud / On-prem / Hybrid Hosting
Main focus of 2005 SOA reference model

Figure 1-2 The API and service economy

Although the integration-centric elements of the original IBM SOA reference model remain important, those elements are not sufficient to address the needs that are arising from the API and *service economy*.

Note: The elements of the original IBM SOA are rendered (in compressed form) as the bottom layer in Figure 1-2. Also, see the following white paper for more information about the IBM service-oriented architecture (SOA) reference model:

ftp://ftp.software.ibm.com/software/soa/pdf/SOA_g224-7540-00_WP_final.pdf

Classic SOA middleware is focused on creating and managing software services, but the other three middleware ingredients of the API and service economy are centered on the following concepts:

- Designing and optimizing a business *persona* through the definition and management of easy-to-use APIs.
- Providing developer portals and marketplace participants to make potential consumers aware of your APIs and the support of onboarding and self-service in a controlled fashion.
- Making the use of APIs as easy as possible, including supporting uniform hybrid composition across multiple providers, environments, and technologies.

Many of these capabilities are well-known in isolation but need to be integrated in new ways. Other elements require fundamental innovation and a different approach to delivery.

2

The composable business

In this time of rapid change, your business needs to be *composable* so that capabilities can be projected and combined as they are needed for any internal or external ecosystem. A composable business drives more engaging applications and processes by seamlessly and intelligently integrating *systems of engagement* with *systems of record*, reaching all the way from the mobile device to the corporate back end.

For more information, see Systems of Engagement and the Enterprise:

`http://www.ibm.com/software/ebusiness/jstart/systemsofengagement/`

More engaging applications and processes intelligently use the context of a business interaction to optimize the experience. Context is crucial not only to optimize the offers that are given to a particular customer, but importantly context is a necessity for making that customer feel like they are being treated as a person rather than just a general business opportunity. Human society is used to people taking into account everything they know about us as a factor in how they choose to interact. A simple example is implied by Figure 2-1 on page 8.

Archived

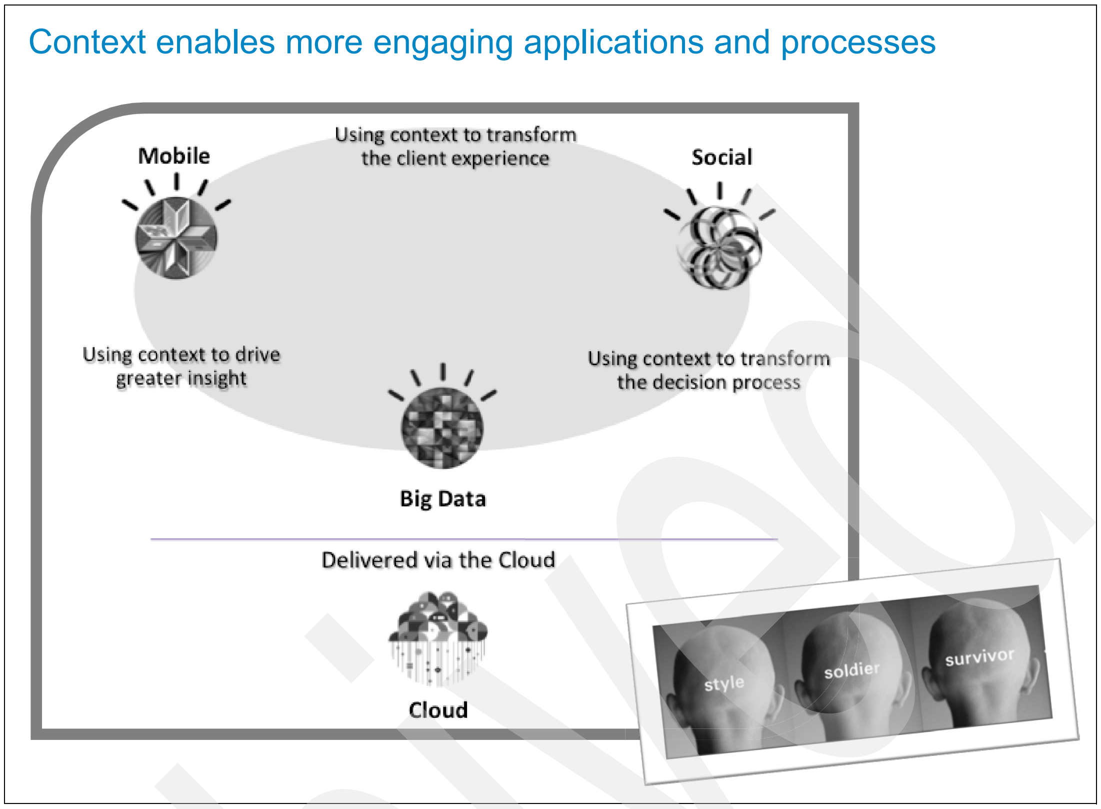

Context enables more engaging applications and processes
Mobile
Using context to transform the client experience
Social
Using context to drive greater insight
Using context to transform the decision process
Big Data
Delivered via the Cloud
Cloud
style
soldier
survivor

Figure 2-1 Engaging applications and processes

You generally do not interact in the same way with a person who is bald by style choice, a person who is a soldier, and a person who is a cancer survivor who recently had chemotherapy. Just knowing their physical appearance is not sufficient; you need to know who they really are to make the optimal choice of how to interact. In the world of software and processes, the confluence of forces changes the way that business operates by allowing you to apply the same type of contextual reasoning that you have always been able to apply in the physical world.

To address how to handle these concepts in practice, this chapter covers the following topics:

- The recipe for a differentiating experience
- SOA design principles and their importance
- API management as the SOA renaissance

Archived

2.1 The recipe for a differentiating experience

As part of the concept of a *composable business*, IBM experts have suggested a recipe for more engaging and innovative business processes (see Figure 2-2).

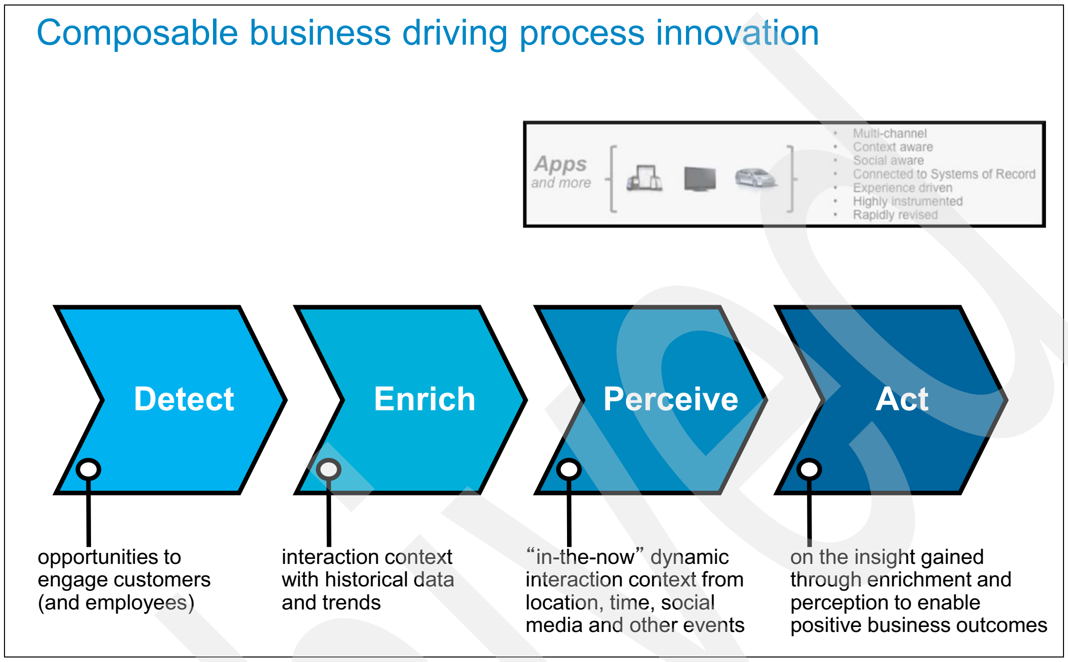

Composable business driving process innovation
Apps
and more
Multi-channel
Context aware
Social aware
Connected to Systems of Record
Experience driven
Highly instrumented
Rapidly revised
Detect
Enrich
Perceive
Act
opportunities to engage customers (and employees)
interaction context with historical data and trends
"in-the-now" dynamic interaction context from location, time, social media and other events
on the insight gained through enrichment and perception to enable positive business outcomes

Figure 2-2 Business process innovation

To cover the four ingredients in the recipe that is shown in Figure 2-2, consider them from a retail perspective:

- **Detect:** A customer is detected walking down the street close to one of your stores.
- **Enrich:** Deepen the understanding of the situation through knowledge of that customer's previous buying behavior.
- **Perceive:** Last night, this customer tweeted about going on a beach vacation soon.
- **Act:** Send an SMS message with a promotion on swim wear.

In the span of a few seconds, you have created a unique and personalized experience. This more personal experience for the customer has improved your chance of generating business and might also have strengthened the long-term relationship with the customer.

As illustrated by this example, the mobile use case deserves special mention. Partially because of the obvious operational challenges in areas, such as security and latency, but even more importantly, because of the business characteristics that are imbued in a differentiating mobile experience. Mobile device users tend to be impatient and always on the move. In fact, the average mobile user spends around 60 seconds using a mobile app before moving on to something else. This means that mobile interactions must be personal (to be relevant) and they must be here and now. Gone are the days where customer segmentation was sufficient to provide so-called personalization. The modern user expects you to know who they are and what they need at this moment. These characteristics are a challenge for modern business systems. However, these characteristics also represent a significant opportunity for enterprises that understand how to embrace them and provide a distinctive customer experience.

Archived

2.2 SOA design principles and their importance

As discussed in Chapter 1, "Business integration in 2014" on page 3, the confluence of forces influences new business agendas and information needs. However, what drives the design to support such business innovation? SOA design principles are a key ingredient in building *composable business* systems that are flexible, robust, and extensible.

Three fundamental aspects are part of and integral to SOA:

- **Service:** A repeatable business task (such as checking customer credit or opening a new account)
- **Service orientation:** A way of thinking about your business through linked services and the outcomes they provide
- **Service-oriented architecture (SOA):** A business-centric architectural approach that is based on service-oriented principles

Figure 2-3 shows a brief outline of these three aspects of SOA.

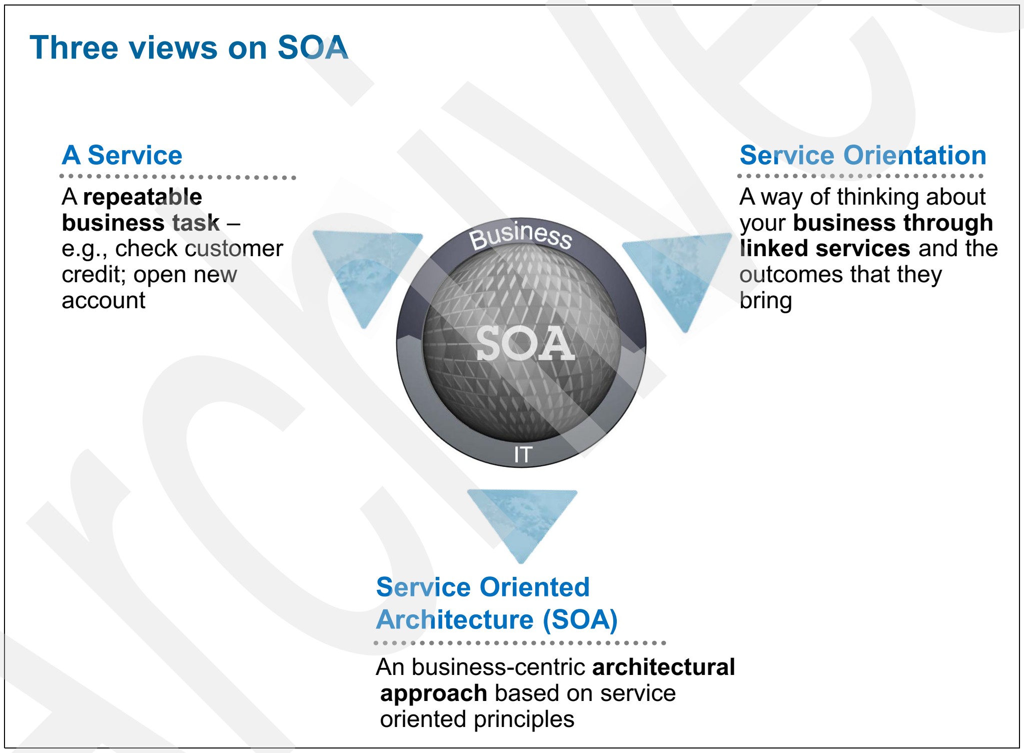

Three views on SOA
A Service
A repeatable business task – e.g., check customer credit; open new account
Service Orientation
A way of thinking about your business through linked services and the outcomes that they bring
Business
SOA
IT
Service Oriented Architecture (SOA)
An business-centric architectural approach based on service oriented principles

Figure 2-3 Understanding SOA

Historically, a lot of focus within SOA has been on service reuse from a software perspective. However, in a business context, the more important notion is that a service is an abstract representation of a repeatable business task.

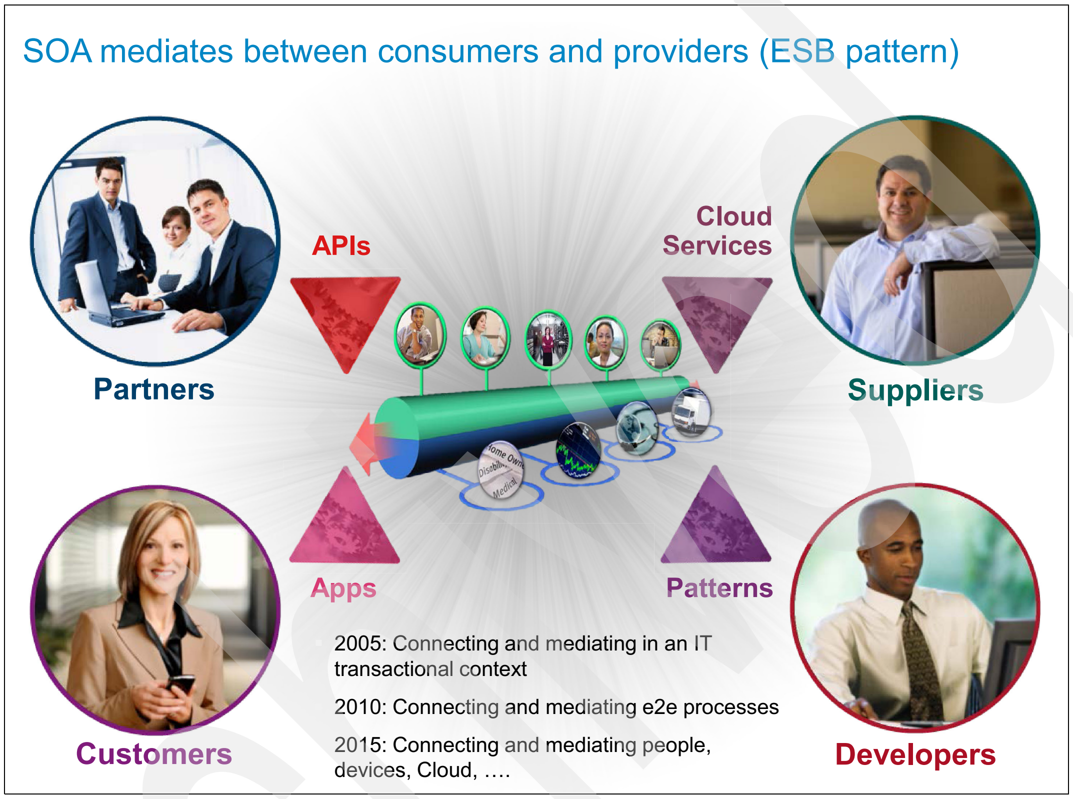
See the Open Group SOA Ontology for the formal standard definition of service:

https://www2.opengroup.org/ogsys/catalog/C144

A service being an abstract representation is important; this allows the service to be projected and accessed beyond the boundary of a physically controlled environment. A service being a representation of a business task is important for designing collaborative business systems above and beyond pure software integration. Finally, the mediation, which is an intrinsic part

Archived

of the enterprise service bus pattern and a fundamental building block of SOA, supports intelligent pairing of consumers and providers of services. This mediation functions whether those consumers and providers are software or people. This latter point is important. In 2005, service mediation was mostly about format and protocol transformation in the context of IT transactions. However, in 2014, business mediation connects people, devices, and processes in an ecosystem that reaches outside the walls of the enterprise (Figure 2-4).


SOA mediates between consumers and providers (ESB pattern)

2005: Connecting and mediating in an IT transactional context

2010: Connecting and mediating e2e processes

2015: Connecting and mediating people, devices, Cloud, ….

Figure 2-4 Intelligent mediation between consumers and providers

The design principles that aid in building such collaborative systems must be broader than the criteria for what constitutes a good service. We believe that the following SOA design principles are fundamental to building composable business systems:

- Service orientation at the core. Thinking about business solutions in terms of interacting processes and services is fundamental.
- Process integration at an Internet scale. Ensuring integrity of interactions and information across time and location is essential.
- Integration with enterprise capabilities and back-end systems. Providing a unified experience across channels and systems and maximizing existing capabilities are essential to drive new innovative processes.
- A basis in industry standards. No single player or vendor can dictate protocols or information standards.
- Providing the platform for a growing ecosystem. As the business ecosystem grows beyond the walls of the enterprise, so does the ecosystem that delivers and manages business solutions.

The last point is what the excitement around web APIs is about: Growing the development ecosystem beyond the enterprise as a way to extend business outreach.

Archived

You can find a more detailed analysis of these SOA design principles and their importance in the ebook titled *SOA Design Principles for Dummies,* which you can download:

http://www.ibm.com/software/solutions/soa/

2.3 API management as the SOA renaissance

When innovation is required, it is tempting to start over with a blank piece of paper. Yet we cannot afford to do this. Although it is true that business models are fundamentally changing, implementation continues to require robust integration of people, processes, and software. But now, it must be at a much grander scale and an even higher speed.

SOA and API management are like *yin* and *yang*, in that neither is whole without the other. API management delivers the business centricity and business model that many SOA initiatives historically lacked, and SOA delivers the experience and engineering discipline that drives good API design and provides robust integration to systems of record. All APIs are services, but not all APIs are good services.

IBM experts believe that API management is the *SOA renaissance*. Far from being an alternative to SOA, API management builds upon and extends the reach of the fundamental principles of SOA. There are many myths about SOA and API management. Chances are good that you have heard people say things such as “SOA is yesterday’s news. APIs are the future,” or “API management is completely different from SOA, and SOA will bog you down.”

From our perspective, this is a classical case of myth-making in the wake of a cool new industry trend. Rarely is a new concept so disruptive that it does not build on the principles and technologies that came before it. With that in mind, consider some of the myths that are specifically related to SOA and API management:

- Myth: “API management is completely different from SOA, and SOA will bog you down.”

  At a technical level, there are more similarities than differences. In fact, at the foundation of most good APIs is a well-designed service. Some of the most important parts of an API are its interface, the business task that it represents, and the associated business contract. Each of these parts is a key element of a service definition.

- Myth: “SOAP is dead. APIs are always REST.”

  It is true that most modern APIs are based on REST and JSON, rather than SOAP. However, this does not mean that there is no use for the formalism of SOAP, merely that such formalism is not necessary for human based consumption of APIs and services. In machine-to-machine communication, or simply when you are using formalized composition tools, there are still plenty of uses for SOAP. Choose the correct binding for the purpose, There are good reasons that people in the IT industry have invented more than one option.

- Myth: “API management is SOA governance rebranded.”

  No, not really. APIs have the nature of products, so they need to be managed as products. SOA governance is less concerned with the treating assets as products than it is with determining and governing the process of creating a good portfolio of reusable assets. In that way, API management and classical SOA governance are highly synergistic.

Archived

- Myth: “No governance is needed with API management, so this allows companies to innovate faster.”

  It might enable you to create things faster, but it also means that you can lose control a lot faster. The APIs that make up your company’s external persona are an important part of your product portfolio and should be managed as such. That said, governance should be lighter for an API than for the typical lifecycle of a back-end software service.

- Myth: “APIs are not versioned.”

  That is like saying that you do not need to change a baby’s diaper. Obviously, disruptive changes do happen, and when they do, you have to handle them. Claiming that versioning is not needed merely means leaving it to API users to figure out the versioning themselves. However, because APIs, in many ways, are business products, we should reduce version churn by coining only a new, official version when an update is not compatible with earlier versions. Therefore, versioning is still necessary, but we must version wisely.

So as we see it, API management is a natural extension of SOA.

API management appropriately refocuses on the business aspects of human and software interactions. Though this was always part and parcel of the idea behind SOA, the business angle, in practice, often got lost in technology. The advent of APIs allows you to separate the business concerns of making an API a successful product, from the IT concerns of providing the service that implements the API. The journey from IT-centric web services to business-centric API management is not merely appropriate; it is necessary for enterprises that build composable business systems that extend beyond the walls of their enterprises.

Good API management solutions provide both the ability to define an API and, more importantly, the ability to project that API into an ecosystem that the enterprise cannot effectively reach through its own end user solutions. Consequently, API management has more focus on the developer experience and the business model, beyond the portfolio of reusable (service) assets.

Archived

Archived

3

Middleware topology of an engaging enterprise

As described in Chapter 2, a *composable business* drives more engaging applications and processes by seamlessly and intelligently integrating systems of engagement with systems of record. This is an integration that crosses the boundary between the controlled enterprise environment and the uncontrollable “Internet of Things.” It reaches all the way from the mobile device to the corporate back end. It is important to realize that direct connections across that boundary are not only inappropriate, they are dangerous. Not only is personal information typically involved, which requires a secure and managed connection, but the traffic that originates outside of the boundary is, by definition, Internet scale, in terms of both volume and spikes. Therefore, traffic needs to be controlled and optimized to prevent bringing down the enterprise IT infrastructure.

In the context of integration throughout and beyond the enterprise, this implies an important distinction among the following topology fundamentals:

- **Messaging:** Moving information payloads from A to B in a reliable fashion
- **Integration bus:** Creating well-structured assets and services that are based on existing data and functions
- **Gateway:** Exposing application programming interfaces (APIs) and services across a boundary in a controlled and optimized fashion

Figure 3-1 on page 16 illustrates the concepts of this integration.

Archived

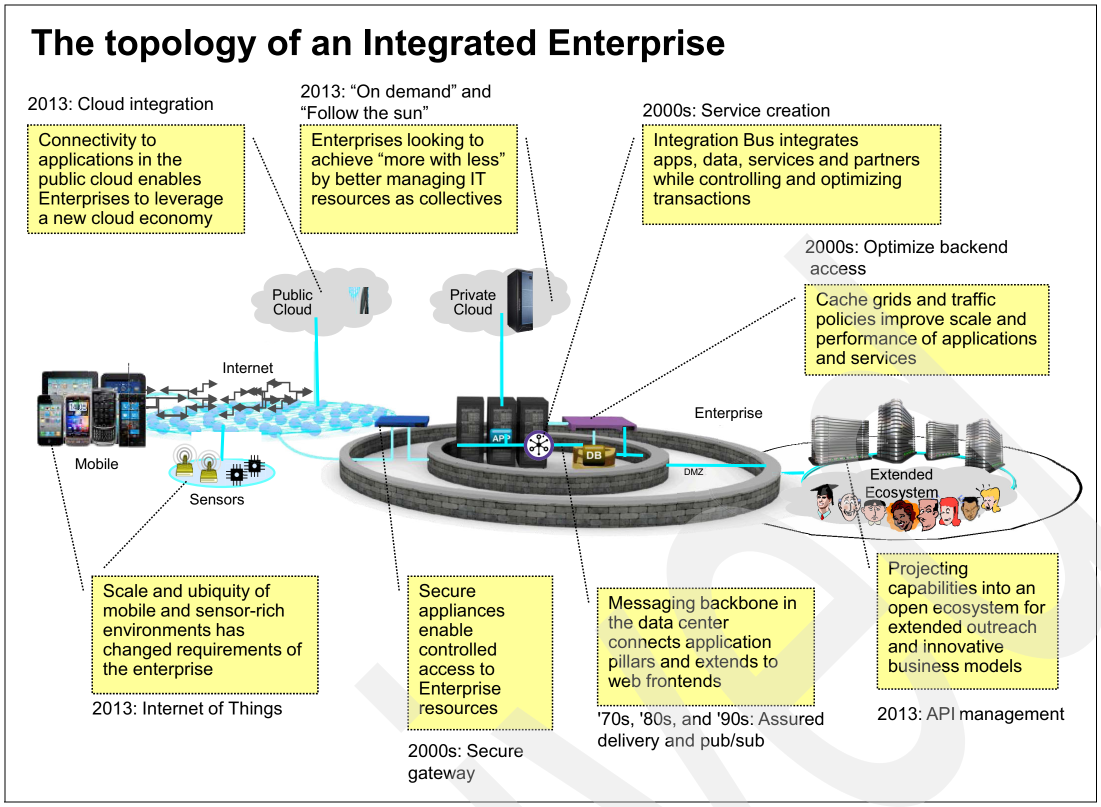

The topology of an Integrated Enterprise
2013: Cloud integration
Connectivity to applications in the public cloud enables Enterprises to leverage a new cloud economy
2013: "On demand" and "Follow the sun"
Enterprises looking to achieve "more with less" by better managing IT resources as collectives
2000s: Service creation
Integration Bus integrates apps, data, services and partners while controlling and optimizing transactions
2000s: Optimize backend access
Cache grids and traffic policies improve scale and performance of applications and services
Public Cloud
Private Cloud
Internet
Mobile
Sensors
Enterprise
APP
DB
DMZ
Extended Ecosystem
Scale and ubiquity of mobile and sensor-rich environments has changed requirements of the enterprise
2013: Internet of Things
Secure appliances enable controlled access to Enterprise resources
2000s: Secure gateway
Messaging backbone in the data center connects application pillars and extends to web frontends
'70s, '80s, and '90s: Assured delivery and pub/sub
Projecting capabilities into an open ecosystem for extended outreach and innovative business models
2013: API management

Figure 3-1 Integrated enterprise

It is important to realize that the boundary between the enterprise and an external ecosystem is not the only boundary of interest. There are many types of boundaries across which traffic must be controlled and optimized, including but not limited to the following list:

- Different owner
- Different security zone (including a DMZ firewall configuration versus the internal one)
- Different operational characteristics (wanted or actual)
- Need for clear and specific *boundary* traffic management control, caching, and so on (often, but not always, rooted in different operational characteristics)
- Different form factor (such as on-premises versus cloud or private versus public cloud)
- Different consumer audience (for example, turning services into API *products* for use by a *foreign* developer community, and developers can be internal or external)
- Different network topology
- Traffic consolidation across a *boundary* (consolidation in terms of consumers, messages, or any other kind of traffic characteristic)

The separation of capabilities into messaging, integration, and gateway forms a fundamental topology that aids in understanding the different roles and purposes of integration middleware products. This topology is also fundamental to the definition of the integration patterns and concepts in Part 2 of this IBM Redbooks publication.

This chapter outlines the formal definitions for these three basic integration topology components and illustrates why each is quite different from the other two.

3.1 Messaging

At the heart of messaging is the ability to move an information payload from origin to destination in a controlled and reliable fashion. Messaging is the most fundamental of the three topology components, although it is not usually categorized as an integration capability. The reason for this is that integration requires messaging, but messaging capabilities can be used for other purposes.

The following list notes some of the key functions of a messaging system from different perspectives:

- Development perspective:
  - Defining the payload structure
  - Creating or consuming payloads
- Runtime perspective:
  - Delivery according to defined quality of service (for example, guaranteed delivery or pub-sub)
  - Optimized routing across the messaging network
- Management and configuration perspective:
  - Defining sender and recipient topologies
  - Defining the quality of service that you want or need

3.2 Integration bus

The core of the integration bus is the ability to create new reusable assets. It is the topology component that is closest to the classical notion of an enterprise service bus. The enterprise service bus (ESB) is, in reality, a general pattern that embodies the service-oriented architecture (SOA) concepts of *consumers* and *providers* that are mediated in a loosely coupled fashion. An integration bus is a particular embodiment of the ESB pattern that is centered around integrating resources within a zone of control. Although an integration bus provides mediation and composition of any conceivable type of resource, it does not provide the advanced security and traffic controls that are necessary for a gateway.

The following list notes some key functions of an integration bus from different perspectives:

- Development perspective:
  - Pre-defined connectors, adapters, and so on
  - Predefined and user-defined integration patterns
  - A composition model and associated tools
  - Code generation
  - Lifecycle and source code management
- Runtime perspective:
  - Continuous availability
  - Near-linear scalability
  - Support for multiple concurrent versions
  - Operational monitoring and insight
  - Workload separation that allows operational reconfiguration without redeployment
  - Routing policy (including support for automatically dealing with environment-specific endpoint addresses)

Archived

- Management and configuration perspective:
  - Dynamic configuration of endpoints
  - Managed promotion between environments
  - Cluster-based workload management

3.3 Gateway

The concept of a *gateway* represents the topology component that is on the boundary between *systems of engagement* and systems of record or any other boundary of interest (see Chapter 2, “The composable business” on page 7). Even though a gateway, by its nature, supports some amount of mediation, it is very different from an integration bus. The difference stems from the fact that the gateway architecture is optimized for control and throughput rather than for aggregation of data and function (which are the focuses of an integration bus).

Technically, a gateway is also an embodiment of the ESB pattern, but gateways have a specialized topology function that is fundamentally different from an integration bus and requires a fundamentally different architecture. Gateways have advanced security and traffic control (a smart and efficient pipe) but no composition and only limited mediation. Gateways are typically limited to a few standard protocols, such as SOAP and REST.

The following list notes some key functions of a gateway from different perspectives:

- Development perspective:
  - Serialized (multiprotocol) transformations
  - API definition and design by proxy or simple assembly (no full-fledged composition)
  - Portal for onboarding, collaboration, developer outreach, and so on (particularly relevant in an API management context)
- Runtime perspective:
  - Continuous availability
  - Near-linear scalability
  - Support for multiple concurrent versions
  - Operational monitoring and insight
  - Workload separation that allows operational reconfiguration without redeployment
  - Automated traffic optimization, including automated caching
  - Opaque proxies (front-side address managed separate from back-side address and no requirements for one-to-one relationships)
  - Machine-to-machine communication (Internet of Things function)
  - Metering
  - Security and traffic management policy
- Management and configuration perspective:
  - Dynamic configuration of endpoints
  - Security management (including the ability to integrate with LDAP, crypto key management, and so on)
  - Traffic policy management
  - Caching policy management

Archived

Archived

Part 2

Integration patterns

This part of the book defines the most common integration patterns and shows how to map these patterns to IBM integration technologies. It covers the following integration patterns:

- Chapter 4, “Enterprise Application Integration pattern” on page 21
- Chapter 5, “Service Integration pattern” on page 31
- Chapter 6, “Gateway pattern” on page 53
- Chapter 7, “Mobile Integration pattern” on page 63
- Chapter 8, “API Management pattern” on page 77

© Copyright IBM Corp. 2014. All rights reserved.

Archived

4

Enterprise Application Integration pattern

This chapter describes the Enterprise Application Integration (EAI) pattern. The EAI pattern is the most classical of the integration patterns that are addressed in this IBM Redbooks publication, and it does exactly what its name says: integrates specific applications. The integration is always directional, in that one application provides a function or data and other applications use it, or *consume* it in service-oriented architecture (SOA) terms. This does not necessarily mean one-way traffic between the applications, because the consumers might need to request responses from the provider.

In all cases, independently of a detailed integration mechanism, the consuming applications are explicitly aware of the providing application.

This chapter covers the following topics:

- Characteristics and capabilities of the pattern
- Ways to implement the pattern with IBM products

Archived

© Copyright IBM Corp. 2014. All rights reserved.

4.1 Characteristics and capabilities of the pattern

The EAI pattern is classically associated with the innermost zone of the topology of the enterprise (Figure 4-1), which is the zone of the back-end applications. The pattern is a direct application-to-application integration approach, and, usually, each end of the integration is aware of the other part. There can be more than one consuming application for a single provider.

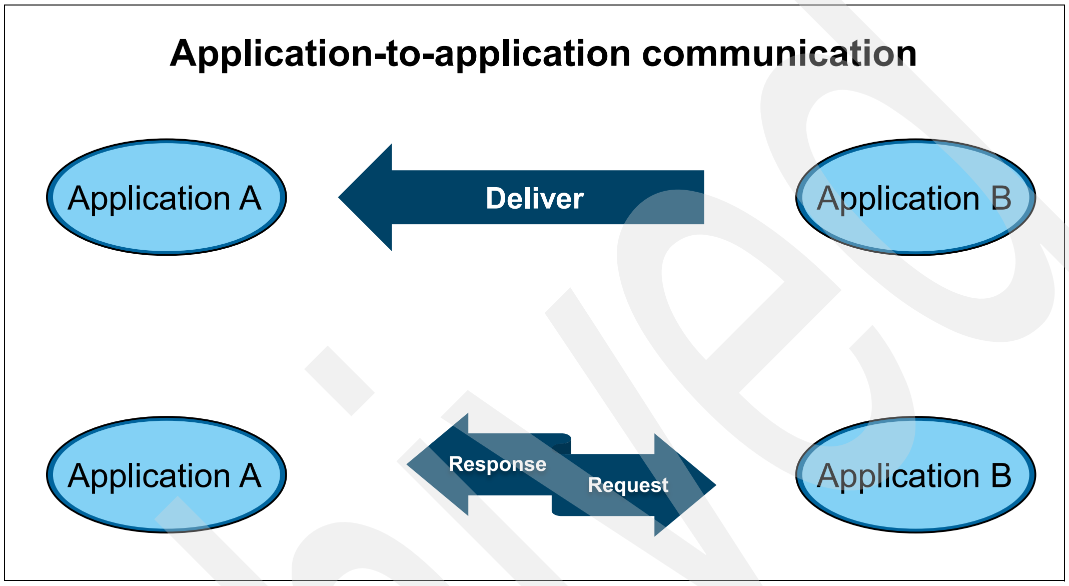


Figure 4-1 General integration patterns, application-to-application communication

Historically, there have been various mechanisms to implement this pattern:

- **Remote procedure-based implementations (such as CORBA):**

  The consuming part of the patterns call the providing part directly. This subpattern can be used only if the two sides of the patterns have a shared procedural interaction form. The transaction function is maintained (or not) according to the characteristics of the procedure call. There is no guaranteed delivery. If the remote procedure call fails, it is up to the caller to figure out how to recover and how to deal with the fact that the receiving end might have only partially processed the procedural call before failing.

- **File-based implementations:**

  The producer part of the pattern creates a data-centric file that the consumer part of the pattern picks up asynchronously. This is especially handy when the applications need to transfer large amounts of data as part of the integration. File-based integration also supports restart in case of operational or data issues. The choice of file format and intermediate storage is part of the proprietary implementation of any particular file-based integration.

- **Message-based implementations:**

  Integration is done through message passing, synchronously or asynchronously. With synchronous communication, both ends of the interaction must be running at the same time. With asynchronous communication, messages are stored outside of the applications until they can be processed. The messaging channels and the formats of the messages are agreed upon by the two parties in the integration. Message-based integration is more immediate than file-based and more reliable than Remote Procedure Invocation. In the context of the EAI pattern, a messaging system plays the role of a universal translator that

Archived

allows communication between any type of applications that are developed in different languages by using multiple technologies and platforms over a period of time. A messaging system also supports different types of transactions, such as *fire and forget*, *guaranteed delivery*, and so on.

In general, the Remote Procedure subpattern is the most brittle of the three, so it is rarely used outside of a single-platform environment. This fragility is what led to the development and success of messaging-based solutions.

The other two subpatterns can be chosen based on the characteristics that you want, as illustrated by Figure 4-2.

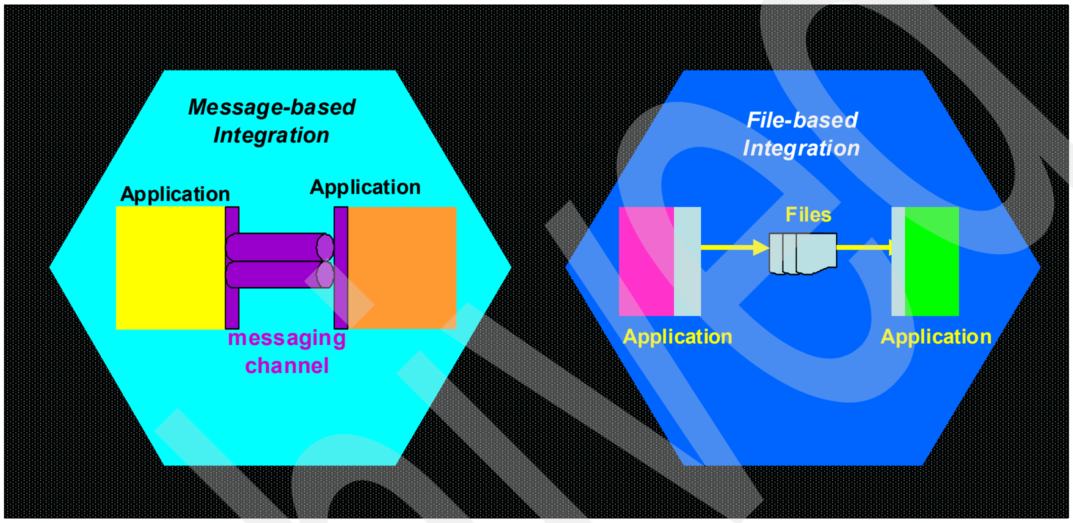


Figure 4-2 Message-based and file-based EAI patterns

The message and file-based subpatterns benefit significantly from middleware in the form of a messaging infrastructure or managed file transfer capabilities. Without such middleware, the two applications that are being integrated need to provide support for transactionality, guaranteed delivery, and so on.

4.1.1 Level of operational control and insight

As described in Part 1: “Topology of an integrated enterprise” of this IBM Redbooks publication, important aspects of building composable business systems are policy-based operational control and the ability to collect insight through sensing of operational interactions. Implementations of the EAI patterns that use messaging or managed file-transfer middleware provide basic monitoring and insight capabilities but little support for policy-based operational control or sense-and-respond style business insight. These more advanced capabilities are usually provided as part of an integration bus or gateway implementation.

Archived

4.1.2 When to use the Enterprise Application Integration pattern

The EAI pattern, when based on middleware, is generally easy to implement. Use this pattern when there are no requirements on operational insight or policy-based operational control and when the applications that are being integrated are known in advance and rarely change. The pattern is useful for integrating different applications that are running within the same back-end environment, but it can be used across a more complex topology if that topology is static. Thinking about the EAI pattern as either messaging or managed file transfer provides a good guideline for when to use it and when not to use it.

In complex topologies that change often, EAI pattern implementations can be expensive to maintain as the number of proprietary point-to-point integrations grows exponentially with the number of applications. Although middleware for messaging and managed file transfer largely mitigate that cost, the EAI pattern does not support the operational characteristics required of many modern systems.

However, many existing integrations are based on the EAI pattern, so and it will continue to be used for a long time. Consequently, it is important to keep its characteristics in mind and choose wisely when to replace an existing EAI integration with something more flexible and when to keep the pattern in place, as-is. Furthermore, newer integration patterns, such as the Service Integration pattern or the Gateway integration pattern, often have messaging capabilities at the heart of their implementations; they are simply adding another layer of higher-level capability on top.

Generally, consider replacing an existing EAI pattern implementation only when there is a need for improved operational control or insight. Beyond that, change the existing implementation only if there are other tangible benefits or needs, such as using the same capability for many different (and possibly unpredictable) consumers.

4.1.3 Retail business example

A typical requirement that a retail appliance store might have is for stock management. Whenever any branch of the store sells an appliance, that information needs to be sent to a central distribution warehouse so that stock levels can be monitored and replacement systems sent to the store.

Although it is desirable that information about the sale is reported quickly, it is much more important that it be reported reliably. So, when everything is working normally, the report gets to the distribution center in real time. But if the network that links the store to the center fails, the report must be held safely and then automatically sent when the network has been repaired.

4.2 Ways to implement the pattern with IBM products

One of the key features of any solution is to remove complexity from application code. Ensuring reliability or recovery is the job of the middleware product or component, such as the messaging provider, so that developers can focus on writing the business logic.

The following list covers some of the factors that should be dealt with by middleware and hidden from application developers, regardless of the business application:

Archived

- **Once-only processing:**

  Business transactions normally happen only once. Middleware needs to ensure that all of the systems that are involved in that transaction do their jobs exactly once. There must be no loss or duplication. Applications must not write complex code for recovery or reverse partially completed operations code (compensation code) if individual systems fail.

- **Ubiquity:**

  Run your applications on the platform where it makes most sense. Be able to integrate applications seamlessly on a wide range of operating systems and hardware environments.

- **Easy to change:**

  Using industry standards can help you become more responsive as skills can be maintained and reused across projects. New applications can be rapidly written and deployed.

- **Easy to extend:**

  Scale your applications rapidly to any volume without downtime. Add more processing without application change. Roll out new applications and features quickly and safely without downtime.

- **Compliance:**

  Meet enterprise, industry, and regulatory requirements easily, especially as those requirements evolve. A middleware implementation should assist with auditing and other compliance objectives to provide a consistent interface for all applications.

- **Performance and availability:**

  Make the best use of the systems. Provide services for high and continuous availability.

- **Security:**

  Provide a secure environment. Data is protected from loss, modification, and reading. Losing sensitive data or failing to comply with regulations costs time, money, and reputation.

These factors are required for many integration patterns. That is why integration solutions are often layered, relying on features of lower levels of the stack to reduce their own complexity.

4.2.1 IBM WebSphere MQ messaging software

IBM WebSphere MQ messaging integration middleware was introduced in 1993 (under the IBM MQSeries name). It has always been focused on providing an available, reliable, scalable, secure, and high-performance transport mechanism to address all of the requirements.

WebSphere MQ has also always connected systems and applications, regardless of platform or environment. It is essential to be able to communicate between a GUI desktop application that is running on Microsoft Windows and an IBM CICS® transaction that is running on an IBM z/OS® system. That value of universality is a core value of the product, and that has not changed. What has changed is the range of environments in which WebSphere MQ is used. As newer platforms, environments, requirements for qualities of service, and messaging patterns arise, WebSphere MQ evolves to meet the needs.

For example, security has become more important as systems are made accessible to more users across an enterprise and beyond. Performance and scalability requirements have increased. Regulators and auditors have imposed more controls on what can or must be done. Systems, which need access to enterprise data, have become both more powerful

Archived

(faster, more processors, and so on) and much less powerful (sensors, tablets, and mobile phones). All of these have been addressed during the existence of WebSphere MQ.

There are now more ways for applications to reach a WebSphere MQ queue manager and get access to any existing applications that were already enabled for WebSphere MQ. New applications can be written that use alternative interfaces and still maximize the reliability and performance of WebSphere MQ. Those new applications do not need a complete replacement of existing infrastructure. The applications work with what you already have and know how to manage.

At the same time as adding new interfaces, protocols, and environments, a large amount of WebSphere MQ workload continues to be run in mainframe-based data centers. Efficient use of the capabilities and integration with the IBM System z® hardware and operating system is critical.

In addition to providing programming interfaces for user-written applications to use, the ubiquity of WebSphere MQ means that there are many prebuilt integration mechanisms and solutions that use its capabilities. For example, the WebSphere adapters make it easy to connect many commonly used enterprise resource planning (ERP) or customer relationship management (CRM) products to other applications that are enabled to use WebSphere MQ.

Other integration patterns in this book also provide links through WebSphere MQ. For example, IBM Integration Bus uses WebSphere MQ internally for several critical operations, as well as exposing endpoints in WebSphere MQ.

For a more complete introduction to messaging concepts and WebSphere MQ, see the WebSphere MQ Primer:

http://www.redbooks.ibm.com/abstracts/redp0021.html?Open

4.2.2 Mapping WebSphere MQ to the EAI pattern

For application developers, IBM WebSphere MQ messaging implements the EAI messaging subpattern by providing simple programming interfaces that isolate the applications from many of the detailed aspects of integration. Data is transferred between applications in messages, where a message includes both control information and arbitrary application data. The structure of the data is defined by the applications, and WebSphere MQ is largely unconcerned with its format or content. The applications communicate through *queues* or *topics*, with names that are known to the interacting participants.

For system administrators, WebSphere MQ gives a consistent model for defining and administering resources and their connectivity. The administrator sets up paths so that messages put in a queue that is used by a producing application on one system are made available to consuming applications that can be running on a different system (see Figure 4-3 on page 27).

Archived

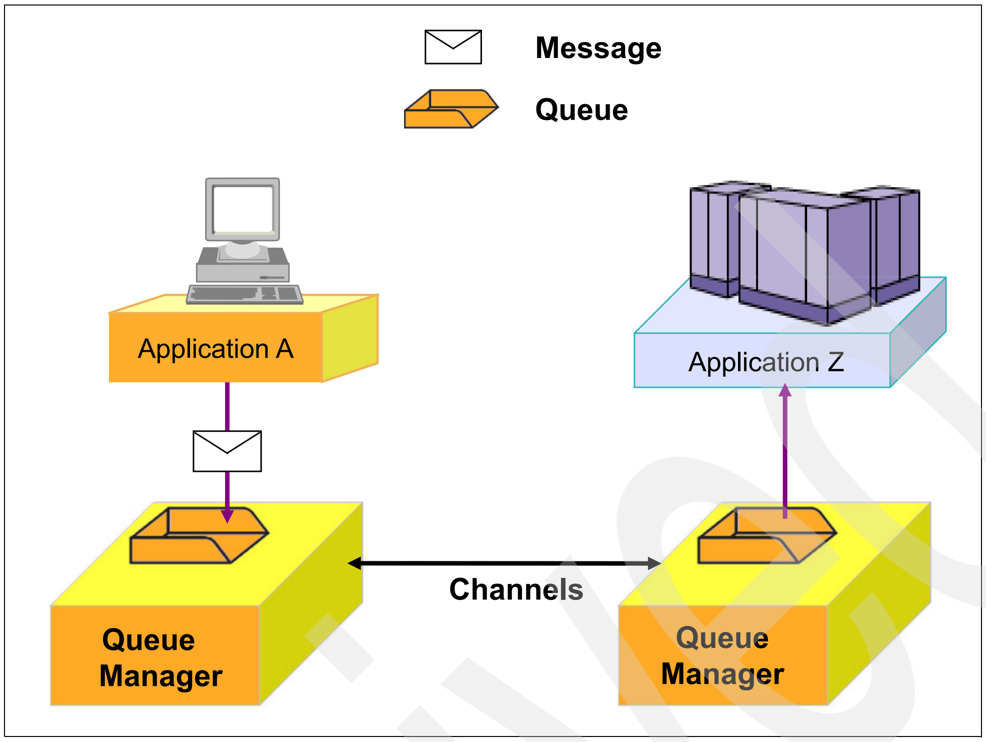


Figure 4-3 Basic WebSphere MQ components to connect applications

Current versions of WebSphere MQ also include a Managed File Transfer (MFT) component to enable the file-based integration subpattern. This MFT capability provides a reliable managed file transfer solution for moving files, regardless of size, between IT systems, which reduces the risk of errors in business data being incorrectly transferred. It offers a reliable, flexible, and cost-effective solution for organizations of all sizes. It uses WebSphere MQ messaging as its reliable transport mechanism for moving files, which enables consolidation of messaging and file transfer infrastructure into a single backbone.

4.2.3 Implementing the retail example

In 4.1.3, "Retail business example" on page 24, the requirement for safe reporting of stock changes was outlined. To implement this with WebSphere MQ, a typical architecture is to have one queue manager at each branch and another queue manager at the distribution center, as Figure 4-4 on page 28 illustrates.

Archived

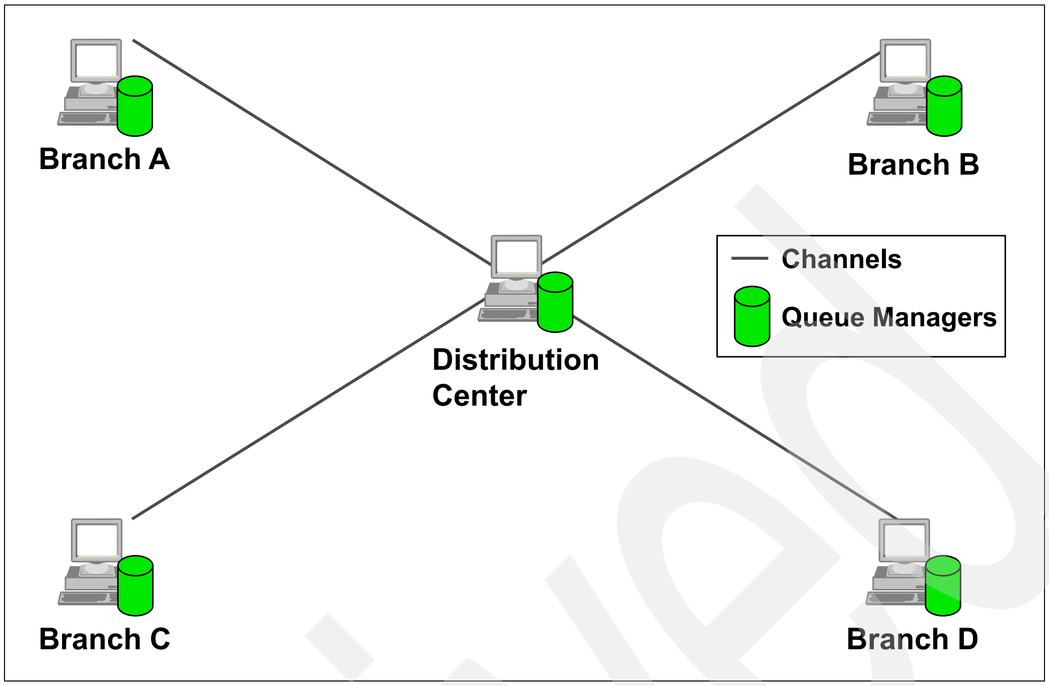


Figure 4-4 A hub and spoke configuration

Each branch has a dedicated channel pair that connects it to the center. There is no need for branches to communicate directly with each other. This topology, which is shown in Figure 4-4, is commonly known as a *hub and spoke* design.

An application that is running at each branch creates a *persistent* message that describes the item that was sold, and **`PUTs`** it into a queue. The application can then immediately return to other work, perhaps creating more messages, without needing to follow the status of the message. If the network is available and the distribution center's queue manager can be contacted, WebSphere MQ immediately transmits the message to the center so that it can be processed by another application. If the network is unavailable, the message is stored reliably for future transmission. This means that even in the event of a power failure that affects the branch store, the message is still available when the queue manager is restarted, so the application does not need to remember that the item was sold.

The application's queue is created so that it points at another queue on the distribution center's queue manager. This is usually done with a **`QREMOTE`** definition. All that the sending application needs to know is the name of this queue. The path to the target system is handled by an administrator.

The administrator of the branch queue manager needs to define a *transmission queue,* where messages are temporarily held until they can be sent to the recipient system, and a *channel* to deal with the network communication. An administrator needs to understand these WebSphere MQ concepts but can then use more advanced features, such as WebSphere MQ clusters, to significantly reduce the number of manual definitions that need to be made as the number of branches grow. WebSphere MQ can automatically define much of what is needed for large-scale topologies.

At the distribution center, another application **`GETs`** the message from its local queue. This application can be a long-running program that waits until new messages arrive or it can be automatically started (triggered) only when there is a message ready.

Archived

For this scenario, it is likely that the central application also needs to update a database, such as an IBM DB2 repository, to hold the stock information. WebSphere MQ makes it easy to write applications that can reliably process both messages and database operations. It supports the use of transactional controls to ensure that, if the application fails part-way through processing, both the WebSphere MQ message and the database updates are kept in sync and restored to a consistent state when the application restarts.

Archived

Archived

Archived

5

Service Integration pattern

With the convergence of mobile communication, social media, cloud services, and big data analytics, the adoption of a business-centric architectural approach that is based on service-oriented principles is important for several reasons:

- Business functionality is externalized through services. Services are logical representations of self-contained and repeatable business tasks that seamlessly integrate systems of engagement with systems of record.
- The interaction between service consumers and service providers is loosely coupled.
- An integration bus provides capabilities to transform, route, or enrich messages between consumers and providers, independent of the platform, data format, and transport protocol.

The Service Integration pattern decouples the interaction between a consumer application and a service provider. It allows multiple consumers to use the services that are offered by one provider even though they know almost nothing about the provider.

By applying service-oriented architecture (SOA) principles and adopting the Service Integration pattern, an enterprise can manage and govern business and IT transformations.

This chapter explains the Service Integration pattern and how the IBM Integration Bus provides the routing, connectivity, and transformation capabilities that are required to decouple service consumers from service providers. It covers the following topics:

- Characteristics and capabilities of the pattern
- Ways to implement the pattern with IBM products

© Copyright IBM Corp. 2014. All rights reserved.

5.1 Characteristics and capabilities of the pattern

The Service Integration pattern is classically associated with service-oriented architecture, where consumer applications and provider applications are decoupled, with these results:

- *Provider* applications encapsulate functions that they make available through an interface.
- *Consumers* start services by using open protocols.

Consumers no longer need connectors (adapters) to communicate with the hub because of the high use of standardized SOA protocols. New requesters can be added at will (although they must be suitably governed).

Traditionally, these services use XML to code and decode data and SOAP to transport it. However, you can use other application and transport protocols, such as JavaScript Object Notation (JSON) to code and decode data and Representational State Transfer (REST) for transport.

There can be more than one consumer application for a single provider. Figure 5-1 shows the main components and how they interact.

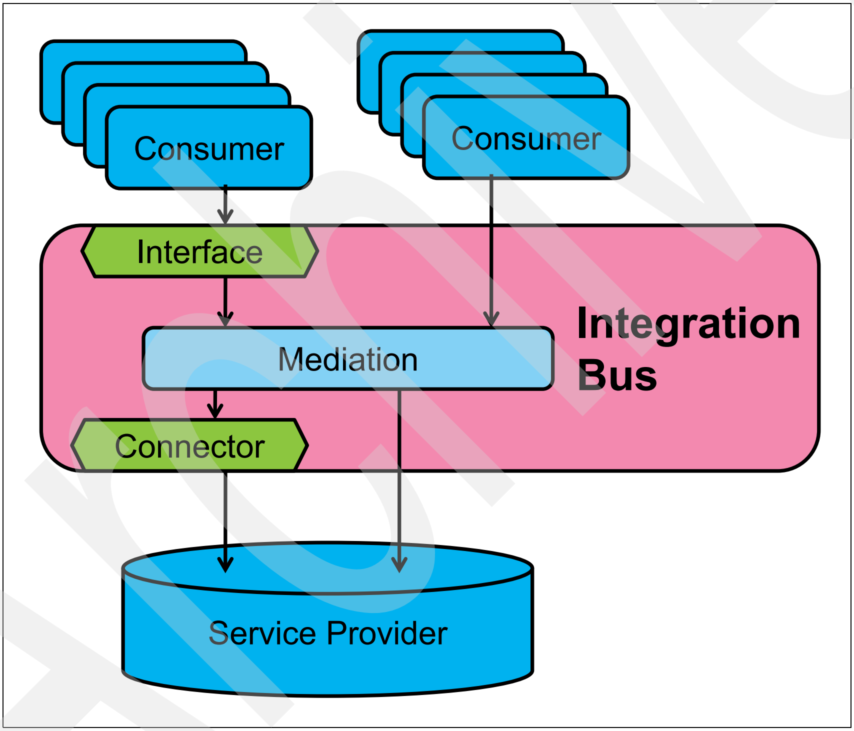


Figure 5-1 Service Integration pattern main components

The service definition provides controlled access to enterprise services over standardized protocols and provides operational and business metrics.

The integration bus provides the capabilities to make the service available (exposed). In addition, it performs traditional integration, such as data mapping, data formatting, and aggregation. It can be used to deliver higher qualities of service by using store-and-forward, retry, error handling, and so on to ensure compliance with the service level agreement (SLA).

Mediation is an intrinsic part of the Service Integration pattern and a fundamental building block of SOA. It supports intelligent pairing of consumers and providers of services.

Archived

Service providers create services and, in some cases, publish their interfaces and access information to a service registry. Optionally, implementers of services can publish service interface definitions to a registry. Consumers retrieve the details for use at development time.

Each provider must decide which services to make available, evaluate trade-offs between security and easy availability, and determine how to price the services or determine how to use the value of the services if they are free. The provider must also decide which category to list the service in and what kinds of trading partner agreements are required to use the service.

Over time, more service providers will make standardized protocols available. But there will always be some older systems that require deeper integration and, therefore, need connectors (also known as adapters).

The integration between consumers and providers is always directional. However, the transport protocol and the data format might not be. One application, which is known as the *provider*, provides functions or data. Other applications, known as *consumers*, use the function or data. Provider applications can also act as consumers of other provider applications.

A *service* is an abstract representation of a business task that can be accessed through a collection of related endpoints. A consumer application connects to a service through an *endpoint*, or *port*. This endpoint describes the contract between the consumer application and the service provider. An endpoint specifies the operations that can be started by a consumer and the binding information (how a consumer can connect to a service). It also provides the network details about where to locate the endpoint. *Operations* describe supported actions by the service. They require an input message, an output message, and optionally, a fault message. The *binding* information provides the protocol name, the invocation style, a service ID, and the encoding for each operation. Bindings are concrete protocol and data format specifications.

There are different styles of interaction between a consumer and a provider:

Synchronous interaction	The consumer requests a response from the provider before it can complete a task. This is also known as a *request-response* interaction.
Synchronous interaction	The consumer requests a response from the provider before it can complete a task. This is also known as a request-response interaction.
Asynchronous interaction	The consumer sends data to a provider but does not require a response to complete a task.

In applying the Service Integration pattern, there are different approaches to security:

- **End-to-end trust:**

  The integration bus trusts the consumer application to authenticate and authorize end users. Service providers consider the hub an authenticated and authorized user. This scenario is frequently used when all components are internal, under the same security management, and part of the trusted zone.

- **Enforce security on application identity:**

  The integration hub does not trust any consumers and requires credentials to validate the identity of any consumer. Provider applications consider the hub as a trusted user. All requests to a provider are made under the identity of the integration hub.

Archived

- **Enforce security on end-user identity:**

  The integration hub applies authentication and authorization. The identity of the end user is required. The provider application considers the hub a trusted user, but it needs the identity of the end user and requires the hub to assert the user’s identity. This model is used where provider services require a secure assertion of the end user identity to provide fine-grained authorization, based on business logic within the provider, or to ensure accountability by securely auditing actions in the provider against the initiating user.

- **Trusted authentication by the consumer application:**

  The consumer application is known and trusted by the integration hub. The consumer application needs to authenticate and assert the identity of the user to the integration hub, which then authorizes the access and asserts the end user identity to the provider application. This model supports single sign-on by users to trusted applications, which can make service requests on their behalf. Identity assertion can be as simple as a user name token but might involve the acquisition of tokens by the requesting application on behalf of a user and subsequent validation of such tokens by the hub.

- **End-to-end authentication, confidentiality, and integrity required:**

  The service provider grants limited trust to the integration hub. The provider requires credentials to authenticate the initiator of the request directly.

  In this model, there is only limited trust on the integration hub by a provider service, which requires credentials that allow the provider to authenticate the initiator of the request directly. In this model, confidentiality and integrity might also be required. This is not a model that assumes no trust in the hub, which is almost certainly trusted to deliver requests to the appropriate provider in accordance with expected service level agreements and to act as a first filter on attempts to access such sensitive services.

  However, the additional levels of end-to-end security, such as digital signatures, are used to identify the initiating user in a way that enables non-repudiation. End-to-end encryption can be used to prevent access by operational staff to highly sensitive information, which might be necessary if exceptions occur in normal processing or when auditing or logging is performed in the hub.

5.1.1 Benefits of adopting the Service Integration pattern

Adoption of the Service Integration pattern provides several benefits:

- You decouple the interaction between consumer applications and service providers. When a service implementation changes, the consumers can continue using the service without modifications. When operational requirements change, you do not need to modify your consumer or your provider applications. However, if the available service interface changes, consumers and the service provider need to be updated to reflect the changes.
- You can write a service once and then reuse it with many applications. Maintenance and testing of the service is reduced.
- You improve interoperability between applications. The integration bus communicates applications regardless of the technology that was used to build it, the transport protocol, or the data format.
- You can make functions available that are hosted in existing applications as services through the integration bus.
- Core integration capabilities, such as routing, message parsing, message transformation, and aggregation, and operational capabilities are provided by the integration bus. Service providers are not required to implement these capabilities within the service. Services deliver business functionality.

Archived

5.1.2 Level of operational control and insight

Operational requirements such as validation, logging, error handling, and others can be implemented by using the integration bus. Consumer applications trust the bus to handle the service's operational requirements. Provider applications focus on the implementation of their functions and delegate operational control tasks to the integration bus.

In the integration bus, you validate the request message before calling a service. Normally, you validate a message against an XML schema. If the message is valid, the service is started; otherwise, the bus returns a data validation error to the consumer application. In adopting the Service Integration pattern, you remove data validation code from your service implementation, which prevents service calls with wrong data. The provider application focuses on service implementation while the bus handles message validation before calling a service.

You might also have requirements to audit data between a consumer and a provider application. Depending on the audit level, you must log different data records. The integration bus offers can log information, which removes the complexity of adding auditing code in your service.

There are different types of errors between a consumer and a provider application that need to be dealt with, such as handling timeout notification errors, message validation errors, security errors, and others. The integration bus offers functions to handle those so that the service focuses on the implementation of what is offered, not on operational aspects.

5.1.3 When to use the Service Integration pattern

The Service Integration pattern is foundational to service-oriented architecture. It is based on a set of architectural principles, patterns, and criteria that address characteristics such as modularity, encapsulation, loose coupling, separation of concerns, reuse, and composability. The Service Integration pattern is centered on integrating resources within a zone of control where advanced security and traffic controls that connect consumers and providers are not required.

Use the Service Integration pattern when you require any of the following architectural capabilities:

- Reuse business functionality inside and outside of your organization to minimize implementation effort and maintenance costs.
- Decouple service implementation from service operational requirements. Consumers have different operational requirements. By using the capabilities of the integration bus, you can implement ad hoc operational requirements that are based on the service level agreements between consumers and the service provider without affecting the service provider implementation.
- Decouple service implementation from binding requirements. Consumers can be hosted on different platforms. They can have different data format requirements and transport protocol requirements. By using the capabilities of the integration bus, you can connect consumers to a service provider without affecting the service provider implementation.
- Provide interoperability between different consumers and providers. You need to add new consumers quickly, regardless of the technology that is used to implement them and their platforms.
- Provide the any-to-any connectivity between consumers and providers within your own company, and beyond your business to connect to your trading partners, by using operational and integration capabilities available through the integration bus.

Archived

- Allow consumers to access functionality that is provided by applications that cannot expose their functionality as a service. You can use the integration bus to expose application functionality in a service-oriented way through mediations.
- Use a programming model complete with standards, tools, and technologies that support web services, REST services, or other kinds of services.

It is best to use the Service Integration pattern whenever you are connecting services and service consumers to achieve decoupling, operational control, and insight.

5.1.4 Retail example

A retail company offers different access channels to customers to buy their products. They provide access on the web, in company stores, and at other stores. Any product that a customer buys can be sent directly from the warehouse or picked up from a store, if it is available. Whenever a product is sold, local stores' stock levels must be updated. The information needs to be sent to a central distribution warehouse so that stock levels can be monitored and kept up-to-date.

Customers who want to buy a product through a web application need up-to-date product availability information from the warehouse. Other vendors need local stock information and, if the product is out of stock, warehouse availability information.

Although it is desirable that information about the sale is reported quickly, it is much more important that it be reported reliably. So, when everything is working normally, the report gets to the distribution center in real time. But if the network that links the store to the center fails, the report must be held safely and then automatically sent when the network has been repaired.

5.2 Ways to implement the pattern with IBM products

The Service Integration pattern embodies the SOA concept of consumers and providers that are mediated in a loosely coupled fashion. The following list covers some of the factors that should be dealt with by the middleware, regardless of the business application:

- **Once-only processing:**

  Business transactions normally happen exactly once. Middleware needs to ensure that all of the systems that are involved in that transaction do their jobs exactly once. There must be no loss or duplication. Applications must not write complex code for recovery or to reverse partially completed operations (compensation code) if individual systems fail.
- **Ubiquity:**

  Run your applications on the platform where it makes most sense. Be able to integrate applications seamlessly on a wide range of operating systems and hardware environments.
- **Easy to change:**

  Using industry standards can help you become more responsive as skills can be maintained and reused across projects. New applications can be rapidly written and deployed.

Archived

Archived

- **Easy to extend:**

  Scale your applications rapidly to any volume without downtime. Add more processing without application change. Roll out new applications and features quickly and safely without downtime.

- **Compliance:**

  Meet enterprise, industry, and regulatory requirements easily, especially as those requirements evolve. A middleware implementation should assist with auditing and other compliance objectives to provide a consistent interface for all applications.

- **Performance and availability:**

  Make the best use of the systems. Provide services for high and continuous availability.

- **Security:**

  Provide a secure environment. Data is protected from loss, modification, and reading. Losing sensitive data or failing to comply with regulations costs time, money, and reputation.

5.2.1 IBM Integration Bus

IBM Integration Bus, formerly known as WebSphere Message Broker, is an enterprise service bus that provides connectivity and universal data transformation for service-oriented architecture (SOA) and non-SOA environments. As one of the core IBM strategic integration products, it is a single-engineered product for Microsoft .NET, Oracle’s Java, and fully heterogeneous integration.

Functions and functionality

The integration bus provides key functions to help you in the adoption of a full, functional enterprise service bus. It is optimized for the integration bus role in the integration topology and enables information that is packaged as messages to flow between different business applications, ranging from large, traditional systems to devices such as sensors on pipelines. It provides a universal integration capability that addresses a wide range of integration scenarios, including web services, such as SOAP and REST, messaging, database, file, ERP systems, mobile and physical devices, email, custom systems, and more (see Figure 5-2).

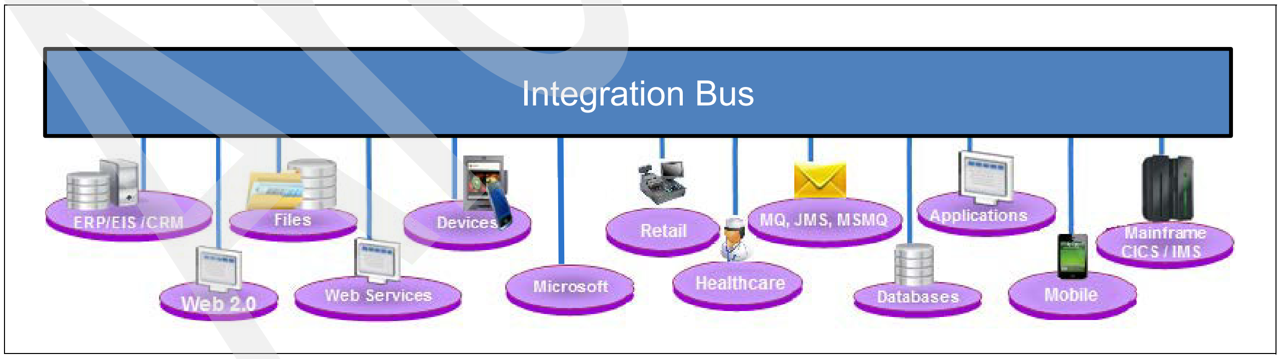


Figure 5-2 IBM Integration Bus

With IBM Integration Bus, organizations of any size can eliminate point-to-point connections and batch processing, which increases business flexibility and interoperability of systems, regardless of platform, protocol, or data format. Their diverse applications can interact and exchange data with other applications in a flexible, dynamic, and extensible infrastructure.

The IBM Integration Bus supports the following protocols, data formats, and operations:

- Protocols:

  IBM WebSphere MQ, Java Message Service 1.1, HTTP and HTTPS, web services (SOAP and REST), file, enterprise information systems (including SAP and Siebel), and TCP/IP.

- Data formats:

  Binary formats (C and COBOL), XML, and industry standards (including SWIFT, EDI, and HIPAA). You can also define your own data formats.

- Operations:

  Routing, transforming, filtering, enriching, monitoring, distribution, collection, correlation, and detection. You can route, transform, and enrich messages from one location to any other location.

IBM Integration Bus patterns

IBM Integration Bus provides patterns (sometimes called *IIB* patterns) that are reusable solutions. They encapsulate a tested approach to solving a common architecture, design, or deployment task in a particular context. A developer can use a pattern to generate customized solutions to a recurring problem in an efficient way. IBM Integration Bus patterns encourage the adoption of preferred techniques in message flow design to produce efficient and reliable flows. These patterns provide the following benefits:

- Guidance for the implementation of common ways to solve similar problems that follow best practices
- Increases development efficiency, because resources are generated from a set of predefined templates
- Results in higher-quality solutions through reuse of assets and common implementation of programming approaches, such as error handling and logging

Archived

For example, you can use the IBM Worklight *Mobile Service* pattern to integrate a mobile application that was written with the Worklight mobile application development platform with a service provider through the IBM Integration Bus (Figure 5-3).

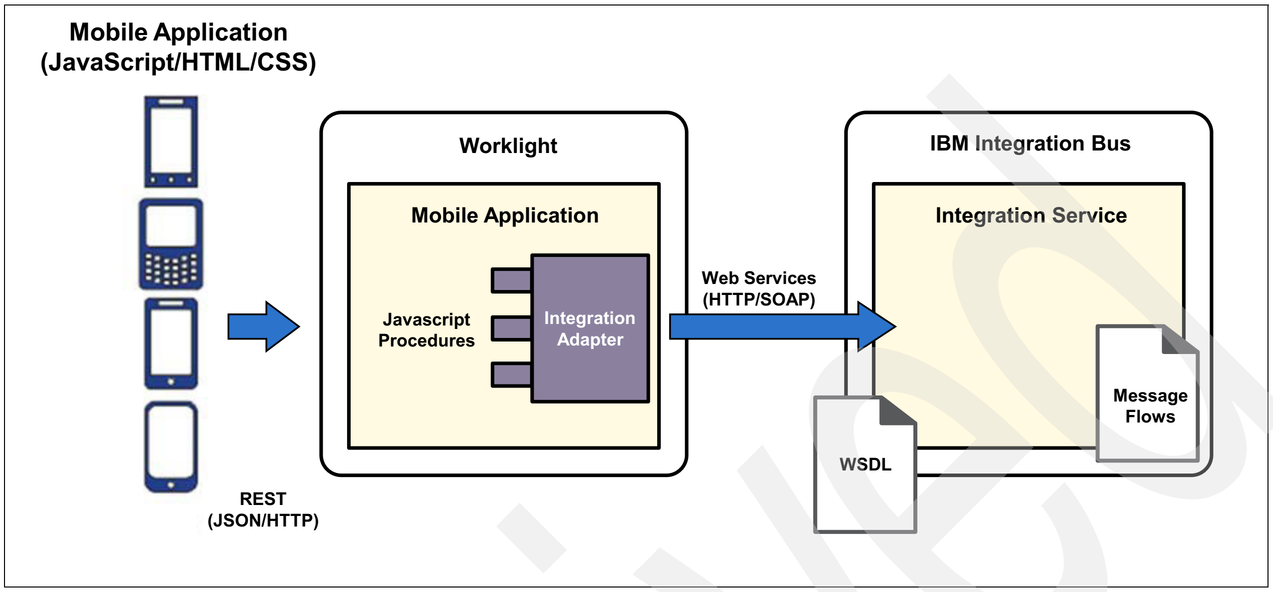


Figure 5-3 Worklight platform to a service provider through the IBM Integration Bus

In another example, you can use the *Service Facade to Microsoft .NET: request-response* pattern to integrate an application that is written for the Microsoft .NET platform by using the IBM Integration Bus. You can also use the pattern to make a .NET class available as a web service by using IBM Integration Bus.

Other supported software and technology

The IBM Integration Bus can interact with a wide range of external systems and resources. It interfaces with these other IBM products and products from other software vendors to provide integration solutions that decouple consumer applications from service providers:

- IBM Tivoli® License Manager
- IBM WebSphere Business Monitor
- IBM WebSphere Integration Developer
- IBM WebSphere MQ File Transfer Edition
- IBM WebSphere Process Server
- IBM WebSphere Service Registry and Repository
- IBM WebSphere Transformation Extender
- Cirtrix Presentation Server
- Enterprise information Systems
  - SAP
  - Siebel
  - PeopleSoft
- Security
  - IBM Tivoli Federated Identity Manager
  - Lightweight Directory Access Protocol (LDAP)
  - IBM Resource Access Control Facility (RACF®) and external security managers (on IBM z/OS systems only)

Archived

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

- Databases and Open Database Connectivity (ODBC) support:
  - IBM DB2 database software
  - IBM Informix®
  - Oracle
  - Sybase
  - Microsoft SQL Server
- Databases and JDBC support
  - IBM DB2 Driver for JDBC and SQLJ
  - IBM Informix JDBC
  - Microsoft SQL Server JDBC driver
  - Oracle JDBC Drivers
  - Sybase jConnect JDBC driver
- File systems
  - FTP
  - SFTP
- Version-control systems
  - IBM Rational® ClearCase® development repositories
  - Concurrent Versions System (CVS)
  - Other repositories that are supported by Eclipse
- Compilers (C and Java)
- Browsers
  - Google Chrome
  - Microsoft Internet Explorer
  - Mozilla Firefox
- Adobe Acrobat, for reading PDF files

The following products and applications that adhere to certain specifications can also interact with the IBM Integration Bus:

- Java and JMS providers
- SOAP
- WS-Addressing
- WSDL
- WS_Security
- XSD
- XSLT

For more information, see *General industry standards supported by IBM Integration Bus*:

http://bit.ly/1g7Wk4O

IBM Integration Bus includes predefined resources that you can configure during the development phase to specify runtime behavior. For example, you can use a TimeoutControl node and a TimeoutNotification node in an application that requires events to occur at a particular time or at regular intervals, such as when you want a batch job to run every day at midnight or you want information about currency exchange rates to be sent to banks at hourly intervals.

Archived

Integrate Bus capabilities

In addition to these resources, IBM Integration Bus provides the following capabilities:

- Workload management

  System administrators can monitor and adjust the operational behavior of a solution. For example, system administrators can control the rate at which messages are processed through manual activities or through policies.

- Timeout notifications

  System administrators can specify and monitor the maximum amount of time that any solution is allowed to wait to process a message and specify an action to be taken if the timeout is exceeded. These can be done through manual processes or through policies.

- Error handling

  Basic error handling is available by default. If basic processing is not sufficient and you want to take specific actions in response to certain error conditions and situations, you can enhance your solutions to customize error handling (see "Error handling options" on page 49).

- Activity logs

  These logs contain information about your integration and how it interacts with external resources, including databases, applications such as SAP and others.

You can get more mediation and operational functions by adding industry packs. For example, the IBM Integration Bus Healthcare Pack, which builds on the IBM Integration Bus, provides support for applications in healthcare environments. This pack includes Audit Trail and Node Authentication (ATNA) audit nodes to create and send audit messages to an ATNA audit repository.

Security management

An important aspect of securing an enterprise system is the ability to protect the information that is passed from external parties. IBM Integration Bus provides key functionality to protect the information that is exchanged between a consumer application and a service provider:

- Secure the transport. For example, you can secure the messaging transport with SSL connections, restrict access to queues, apply WS-Security to web services, and secure access to message flows.
- Secure individual messages, based on their identities.

The security functions that are associated with a mediation are controlled by using *security profiles*, which are created by the system administrator and accessed by the security manager at run time.

A security profile defines the security operations that are to be performed in a mediation at SecurityPEP nodes and security-enabled input and output nodes. A security profile allows the system administrator to specify whether identity and security token propagation, authentication, authorization, and mapping are performed on the identity or security tokens that are associated with messages in the mediation and, if so, which external security provider is used (an external security profile is also known as a *policy decision point*, or *PDP*).

Archived

The following external security providers (policy decision points or PDPs) are supported:

- WS-Trust V1.3-compliant security token servers (including IBM Tivoli Federated Identity Manager V6.2) for authentication, mapping, and authorization
- Tivoli Federated Identity Manager 6.1 for authentication, mapping, and authorization
- Lightweight Directory Access Protocol (LDAP) for authentication and authorization (use the Security Manger, see “Security manager” on page 49)

You can start security in a mediation by configuring either a security-enabled input node or a SecurityPEP node. The SecurityPEP node enables you to start the security manager at any point in the mediation between an input node and an output (or request) node.

Policy sets and bindings define and configure WS-Security requirements that are supported by the IBM Integration Bus for the SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest, and SOAPAsyncResponse nodes. A *policy set* is a container for the WS-Security and WS-RM policy types. A *policy set binding* is associated with a policy set and contains information that is specific to the environment and platform, such as information about keys.

Use policy sets and bindings to define the following items for both request and response SOAP messages:

- Authentication for the following tokens:
  - User name tokens (requires a security profile to specify the external security provider)
  - X.509 certificates (requires the broker keystore and truststore or a security profile to specify the external security provider)
  - Security Assertion Markup Language (SAML) assertions, using SAML 1.1 or 2.0 pass-through (requires a security profile to specify the external security provider)
  - LTPA tokens, using LTPA pass-through (requires a security profile to specify the external security provider)
- Asymmetric encryption (confidentiality) using X.509 certificates (requires the broker keystore and truststore)
- Symmetric encryption (confidentiality) using Kerberos tokens (requires the host to be configured for Kerberos)
- Asymmetric signature (integrity) (requires the broker keystore and truststore)

Either the entire SOAP message body or specific parts of the SOAP message header and body can be encrypted and signed.

5.2.2 Mapping the IBM Integration Bus to the Service Integration pattern

In the IBM Integration Bus, developers can create integration solutions that act as mediations between service providers and service consumers. These solutions can perform typical integration tasks, such as data mapping, data formatting, and data aggregation. They can route requests to different providers based on message content or route responses to different consumers. They can enrich data between a consumer and a provider. They can make functions available as services. They can also handle errors, retry requests, provide logging and auditing capabilities, and more.

Integration solutions are implemented with message flows. These solutions are the building blocks for adopting the Service Integration pattern. They support intelligent integration between consumers and providers of services., and they allow the decoupling of consumers from service providers.

Archived

In the Service Integration pattern, data is transferred between consumer applications and service providers in messages. IBM Integration Bus supports messages of many different data formats over a great variety of transport protocols. A message includes both control information and arbitrary application data. The structure of the data is determined by whether it is a consumer application or a service provider. Consumer applications and service providers can have the same data format, but if they do not, IBM Integration Bus provides capability to receive different types of data over different transport protocols and then transform the message format and transport protocol between consumers and service providers, as needed.

The IBM Integration Bus supports direct connections from applications and can send direct requests to other application endpoints. It can also connect to various subsystems, including WebSphere MQ, files, and databases, to read and write existing application data. You can connect the Integration Bus to your consumer applications or to your service providers by adding the appropriate nodes to your message flow. The nodes that you use can be tailored to support the protocols and subsystems that your applications already use. The Integration Bus supplies nodes to support different protocols and subsystems. You can also create your own nodes to support more protocols and subsystems, if required.

Input nodes

The integration bus provides a broad range of input nodes that a developer can use in a message flow to define the transport protocol and data format for a consumer application. Table 5-1 on page 43 lists some of the input nodes that are built into the IBM Integration Bus.

Table 5-1 input nodes built into the IBM Integration Bus

Input node	Description
SOAP input node	Use the SOAPInput node to process client SOAP messages and to configure the message flow to behave like a SOAP web services provider.
FileInput node	Use a FileInput node if the data that is being processed is stored in files.
FTEInput node	Use the FTEInput node to receive files by using WebSphere MQ File Transfer Edition.
HTTP input node	Use an HTTPInput node if the messages are sent by a REST client or any other type of HTTP client.
JMSInput node	Use a JMSInput node if the messages are sent by a JMS application.
MQInput node	Use an MQInput node if the messages arrive at the broker on a WebSphere MQ queue and the node is to be at the start of a message flow.
.NETInput node	Use the .NETInput node to create inputs for the .NETCompute node by using C#, F#, and VB templates.
TCPIPClientInput or TCPIPServerInput node	Use a TCPIPClientInput node or a TCPIPServerInput node to create a TCP/IP connection when messages are sent through raw TCP/IP sockets.
TCPIPClientReceive or TCPIPServerReceive node	Use a TCPIPClientReceive node or a TCPIPServerReceive node to read the messages that arrive in the message flow through a TCP/IP connection.

Archived

Input node	Description
DatabaseInput node	Use the DatabaseInput node to respond to events in a database.
EmailInput node	Use the EmailInput node to retrieve email, with or without attachments, from an email server that supports Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP).

If the consumer application is an enterprise information system (EIS), a developer can use any of the built-in adapter nodes. The following input nodes are available:

- JDEdwardsInput
- PeopleSoftInput node
- SAPInput node
- SiebelInput node
- TwineballInput node

Nodes for interaction with external systems and resources

You can configure IBM Integration Bus resources to interact with external systems and resources by defining additional resources in your integration solution or mediation. The Integration Bus provides a wide range of nodes that a developer can use in a message flow to define synchronous or asynchronous communications with a service provider. Built-in nodes can also be added, if needed.

For example, the HTTPRequest node interacts with a service by using all or part of the input message as the request that is sent to that service. You can also configure the node to create an output message from the contents of the input message, augmented by the contents of the service response, before you propagate the message to subsequent nodes in the message flow. Depending on the configuration, this node constructs an HTTP or an HTTP over SSL (HTTPS) request from the specified contents of the input message and sends this request to the web service. The node receives the response from the service and parses the response for inclusion in the output tree. The node generates HTTP headers if they are required by your configuration.

The SOAPRequest node is a synchronous request and response node that blocks processing after sending the request until the response is received. This node enables the HTTP 1.1 Keep-Alive method by default.

The SOAPAsyncRequest node can use HTTP or Java Message Service (JMS) transport. It is used in conjunction with the SOAPAsyncresponse node. Both nodes use unique identifies (and, optionally, WS-Addressing) to correlate response messages with the original request. The SOAPAsyncRequest and SOAPAsyncResponse nodes cannot be used with one-way operations. The SOAPAsyncRequest node sends a web service request, but the node does not wait for the associated web service response to be received. This asynchronous functionality enables multiple outbound requests to be made almost in parallel because the outbound request is not blocked waiting for the response. The web service response is received by the SOAPAsyncResponse node, which can be in a separate message flow. The nodes are used as a pair, and correlate responses against the original requests.

The SOAPAsyncRequest node supports two methods of asynchronous requests when using HTTP transport:

- Using WS-Addressing to direct the response to the paired SOAPAsyncResponse node. The SOAPAsyncRequest node waits for the HTTP 202 acknowledgment before continuing with the message flow, and the SOAPAsyncRequest node blocks if the acknowledgment is not received. A new HTTP connection is made by the back-end server to reply to the SOAPAsyncResponse node. This is the default behavior.

Archived

- Using HTTP asynchronous request-response. When the SOAPAsyncRequest property Use HTTP asynchronous request-response is selected, the SOAPAsyncRequest node passes the HTTP socket to the paired SOAPAsyncResponse node to allow the back-end server to reply by using the same socket. When this option is selected, WS-Addressing headers are sent, but they are not required to be understood by the back-end server. The WS-Addressing headers are not marked as `must Understand`, and the replyTo header is set to `anonymous`. Therefore, the node uses asynchronous HTTP socket handling rather than WS-Addressing to make HTTP requests and receive an asynchronous response.

Patterns provided by the IBM Integration Bus

IBM Integration Bus provides patterns, which are reusable solutions that encapsulate a tested approach to solving a common architecture, design, or deployment task in a particular context. A catalog of IBM Integration Bus patterns is provided in the IBM Integration Toolkit, in the development environment. These patterns are divided into pattern categories. The catalog of patterns can contain built-in patterns and might contain user-defined patterns. Built-in patterns cover a set of commonly encountered message flow scenarios and are packaged and released with the IBM Integration Bus. You can also create your own patterns.

Service Virtualization patterns

Developers can use the *Service Virtualization patterns* to provide decoupling between web service consumers and service providers through an IBM Integration Bus mediation. The following Service Virtualization patterns are available (Table 5-2 on page 45):

Table 5-2 Service Virtualization patterns

Pattern name	Description
Service Proxy: static endpoint	Use this pattern to provide decoupling between web service requesters and web service providers by routing through a virtual service that is bound directly to the target service provider.
Service Proxy: static endpoint (web-based)	This is an updated version of the *Service Proxy: static endpoint* pattern, which can be used in the IBM Integration Toolkit and in the web user interface. Use this pattern to provide decoupling between web service requesters and web service providers by routing through a virtual service that is bound directly to the target service provider.

Service Enablement patterns

Developers can use the *Service Enablement patterns* to present consumers with service facades that provide loose coupling between client applications and service providers in timing, protocols, and transport through an IBM Integration Bus mediation. Table 5-3 lists the Service Enablement patterns that are available.

Table 5-3 Service Enablement patterns

Pattern Name	Description
Service Facade to WebSphere MQ: one-way with acknowledgment	Use this pattern to present a web service interface to clients and to fulfil the service requests by using a WebSphere MQ enabled application.
Service Facade to WebSphere MQ: request-response	Use this pattern to provide a web service facade to functions that are accessible only through WebSphere MQ. This pattern creates a bridge between the synchronous HTTP protocol, which is typically used with web services, and existing applications with WebSphere MQ interfaces that cannot be easily upgraded.

Archived

Pattern Name	Description
Service Facade to Microsoft .NET: request-response	Use this pattern to integrate an application that is written for the Microsoft .NET platform with IBM Integration Bus. You can use the pattern to make a .NET class available as a web service by using IBM Integration Bus.
Service Access from WebSphere MQ: one-way	Use this pattern to provide connectivity between a consumer application that handles WebSphere MQ XML messages and a web service provider.

For example, a developer can use the *Service Access pattern* to provide loosely coupled service access to services from applications that are not service-enabled. The Service Access pattern provides the mediations to transform application input into standard web service requests, and also a mediation point for the application of standard functions such as logging (Figure 5-4 on page 46).

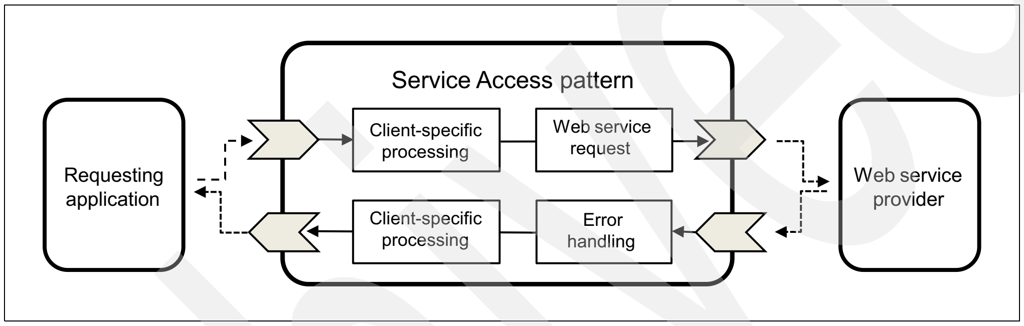


Figure 5-4 Service access

Mobile, file processing, and integration patterns

Table 5-4, Table 5-5, Table 5-6 on page 47,Table 5-7 on page 47, and Table 5-8 on page 48 list some of the integration patterns available in IBM Integration Bus by category. Developers can use these patterns to develop mediations between consumers and service providers by following suggested design approaches when adopting IBM Integration Bus.

Table 5-4 Mobile application patterns

Pattern name	Description
Worklight to Microsoft .NET: request-response	Use this pattern to integrate a mobile application that is written for the Worklight platform with Microsoft .NET applications that are running in IBM Integration Bus.
Worklight: mobile service	Use this pattern to integrate a mobile application that is written for the Worklight platform with a service that is running in IBM Integration Bus.
Worklight: push notification from WebSphere MQ	Use this pattern to send notifications to IBM Worklight mobile applications from IBM WebSphere MQ.
Worklight: resource handler	Use this pattern to provide services to mobile applications that use the Worklight APIs. The services are made available to mobile applications as Worklight adapter procedures that are started from JavaScript in the application.

Archived

Pattern name	Description
IBM MessageSight: Inbound event filter	Use this pattern to configure IBM Integration Bus to receive events from mobile clients through IBM MessageSight messaging appliance, filter the events according to specified criteria, and forward the events that match the criteria to a back-end application.
IBM MessageSight: Outbound event notification	Use this pattern to configure IBM Integration Bus to receive events from a back-end system and dynamically publish those events to mobile clients through MessageSight.

Table 5-5 File processing patterns

Pattern name	Description
Record Distribution to WebSphere MQ: one-way	Use this pattern to bridge between two styles of integration, file-based and transaction-based.

Table 5-6 Application Integration patterns

Pattern name	Description
Data distribution SAP to WebSphere MQ: one way (for IDoc)	Use this pattern to process various types of IDocs that have a single program identifier. When using this pattern, you do not need to redeploy or rediscover existing message sets and adapters, even when you are adding different types of IDocs.
IBM BPM Integration Service	Use this pattern to integrate IBM Business Process Manager Integration services with IBM Integration Bus services. Import a minimal Business Process Manager `export.twx` file into IBM Integration Bus, update the pattern to implement the requested services, and import it into Business Process Manager for final integration.

Table 5-7 Message-based Integration patterns

Pattern name	Description
Message Correlator for WebSphere MQ: request-response with persistence	Use this pattern to accept requests from many client applications on a single queue and to return responses to the correct client by using transactional flows and persistent WebSphere MQ messages.
Message Correlator for WebSphere MQ: request-response without persistence	Use this pattern to accept requests from many client applications on a single queue and to return responses to the correct client by using non-transactional flows and nonpersistent WebSphere MQ messages.
Message Splitter for WebSphere MQ: one way (for XML)	Use this pattern to split a large XML message into smaller elements for processing by one or more targets by using transactional flows and persistent WebSphere MQ messages.

Archived

Table 5-8 Microsoft .NET Integration patterns

Pattern name	Description
Microsoft Dynamics CRM Account Entity output: Static BAPI input	Use this pattern when you want to integrate Microsoft .NET, including Microsoft Dynamics CRM, with IBM Integration Bus. This pattern processes SAP Business Application Programming Interfaces (BAPIs) that are received by a synchronous remote function call and maps them into Create, Retrieve Update, Delete operations on standard Microsoft Dynamics CRM Account entities.
Microsoft Dynamics CRM Account Entity output: Dynamic transport input	Use this pattern when you want to integrate Microsoft .NET, including Microsoft Dynamics CRM, with IBM Integration Bus. This pattern processes user-defined WebSphere MQ, file, or HTTP input messages and maps them into Create, Retrieve, Update, and Delete operations on standard Microsoft Dynamics CRM Account Entities.

5.2.3 Operational capabilities

IBM Integration Bus also provides operational capabilities such as transactionality, workload management, logging, monitoring, error handling, support for policies and security.

Transaction handling

In IBM Integration Bus, developers can configure the transactional behavior of the mediation between a consumer and a service provider. A mediation runs in a single transaction, which is started when data is received by an input node, and can be committed or rolled back when all processing is finished. A developer can configure the nodes in the mediation to determine how the work taken by each node participates in the overall transaction.

- Most nodes for which transactionality is relevant have one or more properties that you can configure to dictate behavior. Therefore, you can decide for each individual node whether it participates in the transaction, or operates independently. Typically, these properties include an option of Automatic so that subsequent nodes in the flow assume the characteristics set by the input node.
- Nodes that support transports that cannot participate in transactions might have other properties to determine how a failure is handled. For example, the FileInput node has a set of Retry properties that you can set to determine failure behavior.
- Nodes that interact with external resources, for which a developer cannot provide properties to configure the transactional behavior, are typically included within the mediation transaction. However, some exceptions exist.

Workload management

The following workload management capabilities in IBM Integration Bus allow system administrators to monitor and adjust the speed that messages are processed and to control the actions that are taken on unresponsive flows and threads:

- Mediation notification

  A common requirement is to be able to monitor the speed at which IBM Integration Bus processes messages. Workload management allows the system administrator to set a notification threshold for individual message flows deployed. An out-of-range notification message is produced if the notification threshold is exceeded. A back-in-range notification message is produced if the notification threshold drops back into range later.

- Setting the maximum rate for a mediation

  The system administrator can set the maximum run rate for an individual message flow. The maximum rate is specified as the total number of input messages processed every second. When set, the number of input messages that are processed across the flow is measured. This measure is regardless of the number of additional instances in use, the number of input nodes in the message flow, or the number of errors that occur. If necessary, a processing delay is introduced to keep the input message processing rate under the maximum flow rate setting.

- Unresponsive mediations

  The system administrator can specify and monitor the maximum amount of time that any message flow is allowed to process a message and specify an action to be taken if the time is exceeded. Manual requests can be made to stop a message flow by restarting the integration server.

  The system administrator can also use policies that are defined within the Integration Registry to configure the workload management capabilities of an integration solution or mediation.

Error handling options

Basic error handling is available by default in any integration solution that is implemented in IBM Integration Bus. If basic processing is not sufficient, and the mediation must take specific actions in response to certain error conditions and situations, a developer can enhance the solution to provide custom error handling. For example, the developer might design a mediation that expects certain errors to be processed in a particular way, or perhaps the mediation updates a database, and in case of error, database updates must be rolled back if other processing does not complete successfully. A developer can adopt either of these approaches:

- Failure checking

  A developer can wire the Failure terminal of a node to explicitly check for any errors that occur within that node. If errors occur, an exception list is propagated to the Failure terminal. The in-flight message remains the same as it was before the node was invoked. The developer can also introduce exit handlers for certain nodes.

- Catching exceptions

  A developer may decide not to wire a Failure terminal. Then, a failure in the node is converted into an exception that is thrown from the node. Any changes that were made to the in-flight message before the exception was thrown are reversed. The exception might cause the current transaction to be rolled back, which means that any updates to transactional resources are reversed. A developer can prevent the transaction from being rolled back, and control the extent to which message changes are reversed, by including a TryCatch node in the mediation.

System administrators can use activity logs to obtain information on how a mediation interacts with external resources, including databases, applications such as SAP, and others. They can also use tags to enrich the log data. *Tags* are strings that represent significant categories of data that can be used for filtering and for providing additional information.

Security manager

IBM Integration Bus provides a security manager, which controls access to individual messages in a mediation by using the identity of the message. A system administrator can configure IBM Integration Bus to process end-to-end an identity that is carried in a message through a mediation by using a security profile. By creating a security profile, you can configure security for a mediation to control access based on the identity that is associated with the message and provide a security mechanism that is independent of both the transport type and message format.

Only a subset of the connectors that are available in IBM Integration Bus use security profiles to control and vary the identity that is used when the connector interacts with an external system. For other connectors, a fixed identity can be specified, which is used to authorize access to the external system. For those connectors, IBM Integration Bus has its own repository of identities, which can be updated if needed.

By default, security capabilities in IBM Integration Bus are based on the security capabilities that are provided by the transport mechanism. IBM Integration Bus processes all messages that are delivered to it, using the service identity as a proxy identity for all message instances. Any identity that is present in the incoming message is ignored.

The following input nodes support mediation security:

- MQInput
- HTTPInput
- SCAInput
- SCAAsyncResponse
- SOAPInput

However, the support for treating security exceptions as normal exceptions is provided by only the MQInput, HTTPInput, SCAInput, and SCAAsyncResponse nodes; it is not available in the SOAPInput node.

The following output nodes support identity propagation:

- MQOutput
- HTTPRequest
- SCARequest
- SCAAsyncRequest
- SOAPRequest
- SOAPAsyncRequest

If the mediation is a web service that is implemented by using the SOAP nodes, the identity can be taken from the WS-Security header tokens that are defined through appropriate Policy sets and bindings. For a SOAPRequest and SOAPAsyncRequest node, an appropriate policy set and bindings can be defined to specify how the token is placed in the WS-Security header (rather than the underlying transport headers).

Use the security manager for these tasks:

- Extract the identity from an inbound message
- Authenticate the identity (by using an external security provider)
- Map the identity to an alternative identity (by using an external security provider)
- Check that either the original identity or the alternative identity is authorized to access the mediation (by using an external security provider)
- Propagate either the original identity or the alternative identity with an outbound message.

5.2.4 Implementing the retail example

In 5.1.4, “Retail example” on page 36, the requirement to support clients who buy products through different channels was outlined. To implement this example with IBM Integration Bus, a typical architecture is to implement one mediation that is used by different consumers. This mediation requests information from a data ware house system. It decouples consumer

Archived

applications from the service provider that is implemented by a data warehouse system, see Figure 5-5.

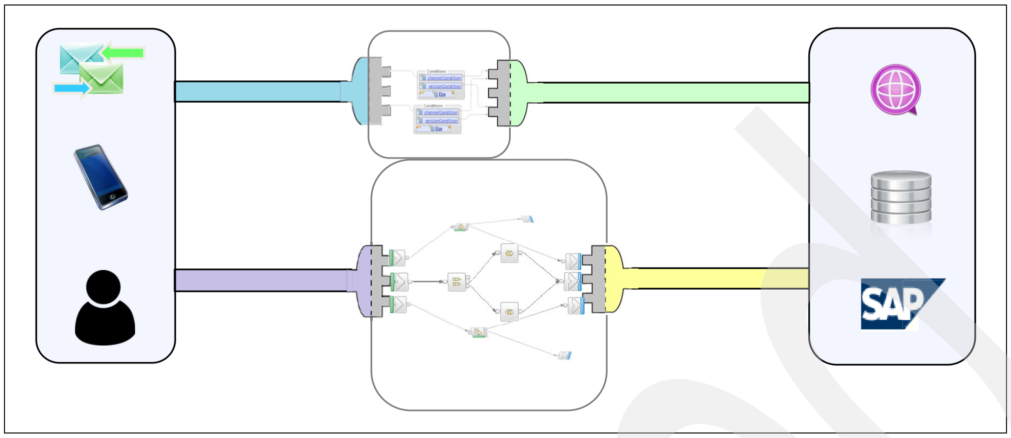
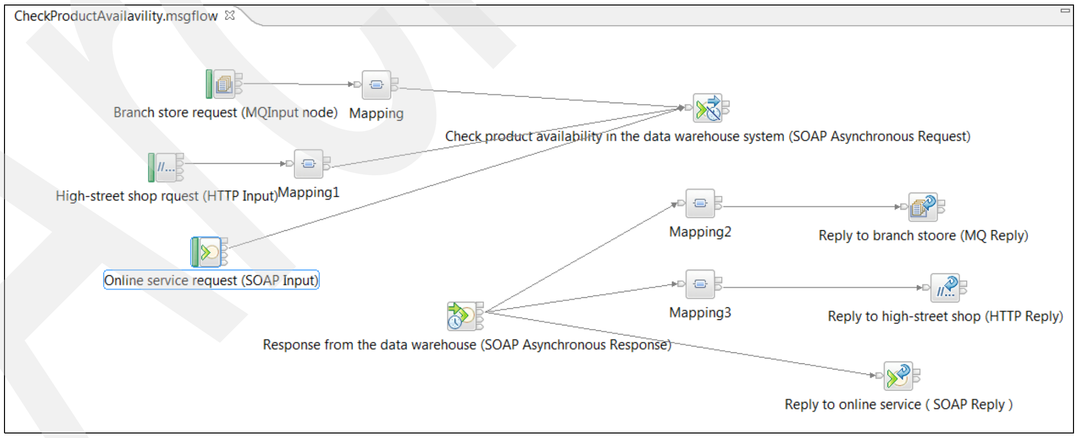


Figure 5-5 A mediation that is used by different consumers

In this scenario, a customer wants to buy a product. He can use an online service or he can go to a store to buy the product. If the product is not available in store, the store must check its availability with the data warehouse system. Whether a customer uses the online service or a shop, each channel requires to check product availability with the data warehouse system at some point in time.

The data warehouse system provides the “Check product availability” service. The implementation is independent of the consumers transport protocol and data formats.

The mediation is implemented as an integration solution in IBM Integration Bus. Multiple consumers can reuse the same service that is provided by the data warehouse system independent of transport and connectivity style. The mediation handles transport and data transformations that are required to access the “Check product availability” service. The mediation decouples consumers from the service provider. See Figure 5-6 on page 51.


CheckProductAvailavility.msgflow
Branch store request (MQInput node)
Mapping
Check product availability in the data warehouse system (SOAP Asynchronous Request)
High-street shop rquest (HTTP Input)
Mapping1
Online service request (SOAP Input)
Mapping2
Reply to branch stoore (MQ Reply)
Mapping3
Reply to high-street shop (HTTP Reply)
Response from the data warehouse (SOAP Asynchronous Response)
Reply to online service (SOAP Reply)
Figure 5-6 Reusing the same service

Data and transport protocol transformation is done with Mapping nodes. A developer can use a mapping to design data transformation graphically.

Archived

The mediation also provides logging capabilities. When the service provider does not respond, the mediation logs information about the failure in an IBM Integration Bus activity log (Figure 5-7).

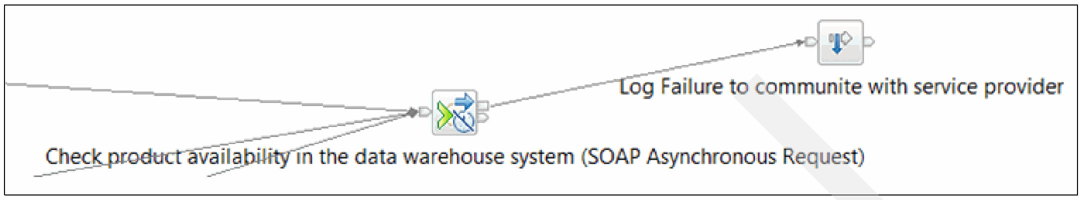

Log Failure to communite with service provider
Check product availability in the data warehouse system (SOAP Asynchronous Request)

Figure 5-7 Mediation logging process

Archived

6

Gateway pattern

This chapter presents an overview of the Gateway pattern from concept to implementation. The following topics are covered:

- Gateway functions and configuration
- Level of operational control and insight
- When to use the Gateway pattern
- Retail example
- Ways to implement the pattern with IBM products

As the name suggests, a *gateway* is a path between two different areas or zones, just as the gate in a fence allows people and things to pass from outside to inside of the fence, and vice versa. In the context of this IBM Redbooks publication, a gateway is a way for information in the form of requests, answers, or updates to pass across any boundary between zones, such as the zone occupied by systems of record (inside the enterprise) and the zones that are occupied by systems of engagement (outside the enterprise).

Archived

6.1 Gateway functions and configuration

The following list notes some of the key functions of a gateway from different perspectives:

- Development perspective:
  - Serialized (multiprotocol) transformations
  - API definition and design by proxy or simple assembly (no full-fledged composition)
  - Portal for onboarding, collaboration, developer outreach
- Runtime perspective:
  - Continuous availability
  - Near-linear scalability
  - Support for multiple concurrent versions
  - Operational monitoring and insight
  - Workload separation, which allows operational reconfiguration without redeployment
  - Automated traffic optimization, including automated caching
  - Opaque proxies (front-side address is managed separate from back side address, and there is no requirements for one-to-one relationships)
  - Machine-to-machine communication (Internet of Things function)
  - Metering
- Management and configuration perspective:
  - Dynamic configuration of endpoints
  - Security management, including the ability to integrate with Lightweight Directory Access Protocol (LDAP), crypto key management, and so on
  - Traffic policy management
  - Caching policy management

DMZ firewall configuration

The gateway provides control and security-oriented services to enable the retail business to offer global customers access to the latest information and facilitates the purchase transaction. The gateway can be used to cache frequently requested resources, which improves performance and lowers compute demands.

The gateway can also serve as a B2B gateway to enable smooth exchanges with a system of trading partners. The IBM DataPower XB62 B2B gateway can perform all of the functions that are described here.

Inner gateway

The inner gateway provides routing, caching, and content translation services so that necessary information flows between disparate systems. This gateway can also balance traffic loads between back-end systems.

Archived

6.2 Characteristics and capabilities of the pattern

The Gateway pattern provides a set of services with the following high-level capabilities:

- Provide secure transport of data (terminating and initiating secure transport layers)
- Interconnect disparate transports (HTTP to IBM WebSphere MQ, for example)
- Implement a service facade (obscure back-end resources, translate SOA-REST, for example)
- Filter information flow (enforce API compliance)
- Improve performance (caching, offloading)
- Route and manage traffic (load balancing and policy-based routing)
- Authenticate and authorize requests (a fundamental gatekeeper function)
- Reflect and enforce service level agreements (SLAs), especially for policy-based configuration and runtime behaviors
- Provide activity and performance management data (real-time insight)

Figure 6-1 is an illustration of a typical flow between clients and services through the gateway.

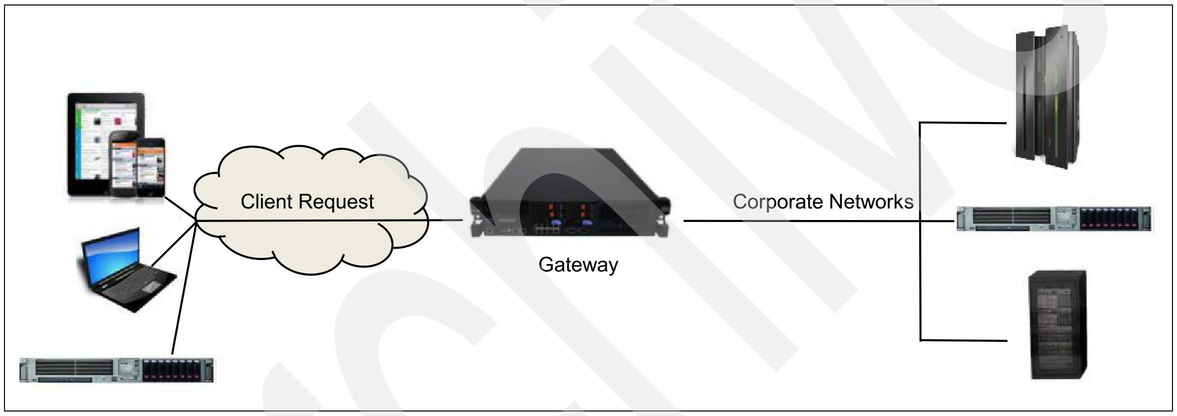


Figure 6-1 Gateway pattern workflow

Clients initiate requests at any time and from any location that can reach the gateway. The gateway can be in the DMZ firewall configuration, or it can be on the boundary between two zones, one of which might not be the Internet. The gateway provides the services that are listed above and discussed further below. The gateway then forwards these requests to the desired destinations, which are typically locations within the enterprise network. Responses, if any, then flow back through the gateway to the requesting clients.

These capabilities offer several benefits:

- Increases ease of connection, which allows the following flexibility for clients:
  - Use their existing transport protocols
  - Use their existing data formats
  - Use Web Services Description Language (WSDL) and other metadata to interpret message exchange patterns

Archived

- Improves performance of existing and new resources by using the following methods:
  - Filters out incorrect or malicious data
  - Offloads compute-intensive tasks, such as cryptography and validation
  - Intelligently routes traffic to appropriate application resources
  - Caches frequently requested data to free back-end resources
- Safeguards valuable internal resources in the following ways:
  - Obscures location of internal resources
  - Employs advanced cryptography for data privacy
  - Enforces authentication and authorization of client requests
  - Enforces SLAs to prevent overload attacks
- Monitors and responds to actual traffic flows with the following actions:
  - Generates a real-time stream of measurement data
  - Logs the chosen activity
  - Accepts and act on SLA updates
  - Routes around bottlenecks or unavailable resources

6.2.1 Level of operational control and insight

The Gateway pattern provides a high degree of operational control. It enables administrators to determine what traffic is allowed to pass across a boundary, according to a wide and deep range of conditions. As a boundary-oriented pattern, this degree of control is required.

The pattern also provides insight into the traffic that is passing through the gateway. Although the gateway itself might not deliver aggregated metrics and reports, external programs that can and do provide such information can easily get the necessary data from the gateway in real time by using any of several different methods, such as Simple Network Management Protocol (SNMP). These metrics can be used to establish or adjust service levels, which the gateway can then enforce. Such enforcement also provides protection for the systems behind the gateway.

6.2.2 When to use the Gateway pattern

The Gateway pattern is useful whenever and wherever you have a boundary that requires security or traffic control. (See Chapter 2, “The composable business” on page 7.) Certainly, security and traffic control are required at the boundary between the enterprise and the open Internet, or even between the enterprise and trusted business partners. However, it is also useful to institute some kind of traffic control between core systems of record and systems of engagement, both of which might be within the boundaries of the enterprise, to protect those systems of record from risky volumes of traffic.

Because the Gateway pattern can be used to increase performance through caching, this pattern can be useful at many different kinds of boundaries, with or without a combination of security or control-related functions.

Archived

6.2.3 Retail example

Consider this retail business scenario: A retail business needs to use technology to increase revenue but also needs to lower or contain costs. Therefore, that business is likely to have the following requirements:

- Increase ease of access to product information and buying channels
- Report real-time availability of products
- Manage inventory and services
- Interconnect with suppliers and banks

The Gateway pattern can play a positive or even an enabling role in each of these requirements. They are positioned in the solution architecture as illustrated in Figure 6-2 on page 57.

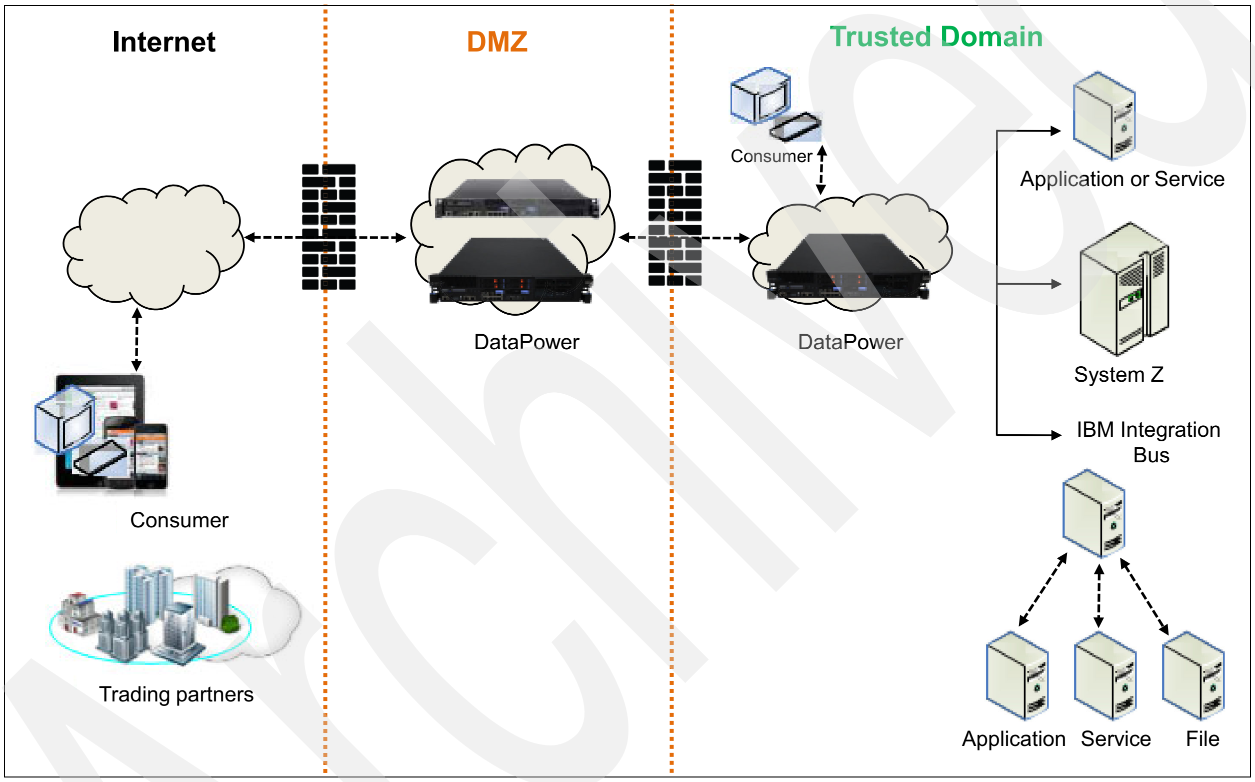


Figure 6-2 Example architecture

6.3 Ways to implement the pattern with IBM products

The IBM WebSphere DataPower Integration Appliance XB62, XI52, and XG45 are purpose-built hardware platforms for delivering rapid data transformations for cloud and mobile applications, secured and scalable business integration, and an edge-of-network security gateway in a single "drop-in" appliance.

Virtual editions of these appliances are also available for use in development and test environments. The virtual appliances offer all of the same capabilities as the hardware appliance but with the flexibility of virtual instantiation.

Archived

The following subsections highlight the different capabilities of these appliances so that, when you decide to implement a gateway in your enterprise, you can identify which one is most suitable for your business.

6.3.1 Security

The DataPower service-oriented architecture (SOA) appliances implement security, particularly in the hostile environment of a DMZ firewall. The hardware and firmware are optimized to work together, so there is no separation between operating system and applications. It is a single firmware image. The default behavior of the device is to disallow access. Starting from this baseline, the IBM WebSphere DataPower device offers the following security-oriented capabilities:

- **Service facade:**

  After it is in place, the DataPower device stops all direct access to back-end application resources. Clients connect to the gateway instead. The address, port numbers, and routes to back-end services are all masked. If you prefer, the gateway can also change data formats to eliminate any external detection of internal and proprietary data formats.

- **Transport Layer Security:**

  DataPower can both initiate and terminate Transport Layer Security (TLS) connections, so it off-loads the computationally expensive cryptography to the device. DataPower can use SSL3, TLS1.0, TLS 1.1. and TLS 1.2 protocols and can bridge between them. A full set of cryptographic algorithms is available.

- **Validation:**

  DataPower can automatically validate data formats, such as XML, SOAP, and JavaScript Object Notation (JSON). In addition, custom schema validation can be applied to any message flows, effectively offloading error processing from the back-end applications.

- **Privacy:**

  The hardware-accelerated cryptographic modules of the appliance can encrypt and decrypt payload data at nearly wire speeds. Both existing and updated cryptographic algorithms are supported. This includes full support for public key infrastructure (PKI) encryption and symmetric key encryption. Even if the message payload is unwrapped from the protection of Transport Layer Security, the payload still cannot be read because it is encrypted.

- **Integrity:**

  To ensure the integrity of payload data, DataPower can employ digital signature technology to both create and verify digital signatures. DataPower offers several methods for signing XML-formatted payloads (XML or SOAP) and can sign, verify, or sign and verify binary data payloads.

- **Authentication and authorization:**

  Using a highly flexible authentication and authorization subsystem known as authentication, authorization, and auditing (AAA), DataPower can easily handle fundamental gatekeeper functions. Also, the AAA system can act as a fluid token changer, accepting one form of authentication and changing it to another to interoperate with disparate or existing authentication systems. For example, clients can employ a Security Assertion Markup Language (SAML) token to assert identity while the enterprise uses a Lightweight Directory Access Protocol (LDAP) systems for authentication and authorization. At the same time, back-end systems might require a Lightweight Third Party Authentication (LTPA) token to grant service. The DataPower device can handle this scenario with no custom coding.

DataPower includes support for authentication systems that are used by modern application programming interface (API) and mobile architectures, such as OAuth 2.0 and Extensible Access Control Markup Language (XACML).

- **Integration with enterprise systems:**

  As already mentioned, DataPower can act as a token changer, which enables integration with various authentication systems, such as Kerberos, LDAP, Netegrity, SAML, OAuth, and IBM Tivoli Federated Information Manager software.

6.3.2 Interconnection

The DataPower device can connect, or *bridge*, disparate systems in several different ways. Serving as a gateway allows the device to accept inbound requests over a client's preferred transport protocol and then to send these requests over a different protocol to back-end systems inside the enterprise. Similarly, the device can translate between data formats if you want or need that translation.

Protocol bridging

DataPower can use HTTP or HTTPS, MQ, FTP or FTPS, EMS, or NFS transport protocols on both the front and back sides of the device, which allows each side to be different if you prefer. In addition to the significant flexibility that this offers, the device can offload the work that is required to initiate and terminate secure transport connections to other systems (HTTPS or FTPS).

Data format bridging

It is not unusual in modern architectures to encounter the need to bridge between REST-based systems that send JSON documents and SOA-based systems that use XML documents. DataPower can bridge these protocols and data formats easily. DataPower also offers translation between XML and binary formats (such as COBOL Copybook) and the reverse. Using the IBM WebSphere Transformation Extender module in DataPower allows any-to-any data format translation.

6.3.3 Routing

The DataPower device can route traffic by using any of these mechanisms:

- **Content-based routing:**

  DataPower can examine the transport headers, the content headers, or the content of messages to make routing decisions. The device can also refer to external resources, such as IBM WebSphere Service Registry and Repository, to get routing information. This allows a gateway to dynamically route traffic to endpoints, for example, sending purchase requests that are USD 10,000 or more to one application and all others elsewhere.

- **Intelligent load distribution:**

  By using the Application Optimization feature, DataPower routes traffic to intelligently distribute loads to back-end clusters and can balance loads among a set of DataPower devices. This capability can potentially eliminate a tier in enterprise architectures.

- **Versioning:**

  DataPower flexible routing capability eases the management of back-end application versioning and front-end API-style versioning.

Archived

6.3.4 Caching

The DataPower device implements data caching in two different ways: on-device caching and off-device caching.

On-device caching

The device can dedicate a user-defined amount of memory for data caching. This caching relies on the establishment of rules. When a response from a back-end application meets the caching criteria, the response is cached. Any future requests processed by the gateway return the cached response rather than forwarding the request to the back-end system. These cache entries can time out, and they require a periodic request to the application. This keeps cached entries sufficiently fresh and relevant.

Off-device caching

The device can use a side cache that is outside of the device, which is typically the DataPower XC10 caching device. A set of rules that are applied to messages that are passing through the gateway determines what responses are cached by the off-device XC10 appliance. Any future requests processed by the gateway return the cached response rather than forwarding the request to the back-end system. As with on-device caching, these cache entries can time out, and they require a periodic request to the application. This helps keep cached entries sufficiently fresh and relevant.

6.3.5 Management

The DataPower device offers several methods of managing the gateway. The method that is used depends on what you want to manage and how. In this area, DataPower can work seamlessly with other IBM products, such as WebSphere Service Registry and Repository, or with third-party products, such as SNMP monitors.

Configuration

Administrators can use any of the following methods for managing the configuration of the device as a whole or for only selected parts, such as application domains that contain one or more services or just individual services:

- Web GUI, the standard browser-based interface
- XML Management interface allows administrators to send XML-formatted files to the device to get or set configurations
- WebSphere Appliance Management Center provides an off-device centralized administration console for multiple devices
- WebSphere Service Registry and Repository and DataPower services can obtain configuration data that is stored in this repository on bootup or during run time. WebSphere Service Registry and Repository also provides an advanced governance framework for managing DataPower devices

Service levels

DataPower includes built-in SLA and service level management (SLM) capabilities, which can shape or throttle traffic, as needed. These settings can also be effectively managed from WebSphere Service Registry and Repository by using the WS-MediationPolicy standard. This allows DataPower to act as an enforcement point for service level agreements.

Archived

Metrics

Administrators can get activity and performance data in real time from the DataPower device by using SNMP, WS-Management, or Web Services Distributed Management (WSDM) protocols or by polling the device through the XML Management interface. Integration with IBM Tivoli Composite Application Manager management consoles is seamless. Any SNMP monitor product can be used.

Logging

Logging information is critical to troubleshooting, and the DataPower device can be set to a debug level of logging for deep inspection when needed. In production environments, the device can simply log critical errors and send those logs from the device automatically. More urgent failure notification is also available and easy to configure. Some use cases require logging of actual messages, such as consumer credit banking, which the DataPower device handles with a special action that captures and logs these payloads. The device integrates with centralized logging systems by using HTTP, FTP, or syslog-based delivery.

Archived

Archived

7

Mobile Integration pattern

This chapter describes the mobile application integration with enterprise back-end services. The *enterprise* can be a bank, other financial institution, retail store, hospital, and so on that provides services to people or the external businesses. In our fast-moving world, each of us wants to connect to these businesses quickly, and mobile devices are one of the fastest channels to connect with them.

We need a faster and dynamic way to integrate the enterprise services with the mobile applications. The integration of these services might involve a request-response between a mobile device and the enterprise or a push notification from the enterprise to the mobile device users. We need a Mobile Integration pattern to reuse in such scenarios.

The following topics are covered:

- Characteristics and capabilities of the pattern
- Ways to implement these patterns with IBM products
- Implementing the mobile banking example

Archived

© Copyright IBM Corp. 2014. All rights reserved.

7.1 Characteristics and capabilities of the pattern

There are two main types of mobile application integration pattern scenarios, which are shown in Figure 7-1:

- A mobile application sends a request to an enterprise system and gets a response.
- An enterprise pushes a notification message to a mobile application.

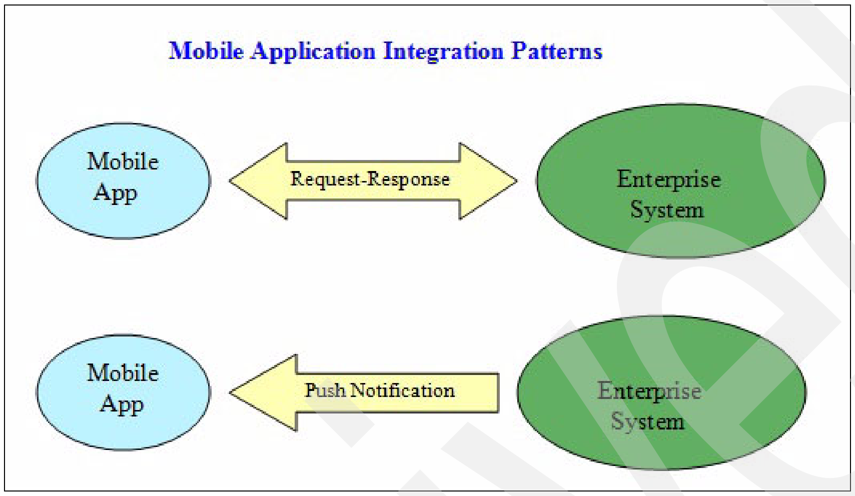


Figure 7-1 Mobile-to-enterprise system integration, Mobile Integration pattern

7.1.1 Defining a request-response pattern

To define a request-response pattern, mobile-ready interfaces are provided for the enterprise services that are created in the enterprise service bus (ESB). The pattern provides the mobile enablement of the integration services in enterprise service bus (ESB). The mobile integration adapter helps to integrate the mobile application with the enterprise services that are running in the enterprise service bus, as shown in Figure 7-2 on page 65. That process involves these components and steps:

1. The mobile adapter sends the inbound request straight to the enterprise service.
2. The enterprise service in the ESB connects to the required back-end systems to process the inbound request through the back-end system and gets the response from the back end.
3. The mobile adapter sends the enterprise service back-end response to the mobile application.

Optional: A gateway can be placed between the mobile application and the Mobile Integration adapter.

Archived

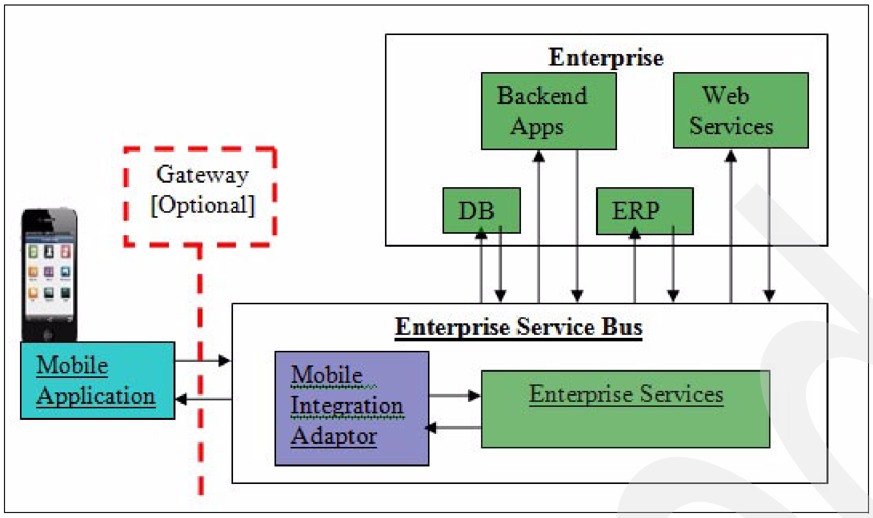


Figure 7-2 Request Response: Mobile Service Integration pattern

7.1.2 Defining a Push Notification pattern

In the push notification scenario, the enterprise application pushes the notification messages to the mobile devices. It uses a mobile back-end service that is running on the mobile integration server, although the notifications can be either polled from the back end or pushed by the enterprise application.

The enterprise application pushes the device notification messages by using the *put* notification service in the ESB, and then the ESB calls the mobile back-end service to push notification messages to the mobile devices that are subscribed for that specific event source notification. The notification is received by the mobile devices regardless of whether the application is running. The notification can be an alert message, badge message, sound, and so on.

The workflow diagram in Figure 7-3 on page 66 shows more detail about these basic steps for creating the Push Notification pattern:

1. The enterprise application pushes a notification message to the mobile device, using the ESB and mobile back-end service.
2. The ESB receives the notification and calls the mobile back-end service to push messages to go to the mobile device.
3. Messages go through the mobile integration server (back-end service for push notification) to the mobile application and the mobile device.

For more information about the mobile push notification implementation, see the link to “Using Worklight API for push notifications in native iOS applications” at the end of this chapter.

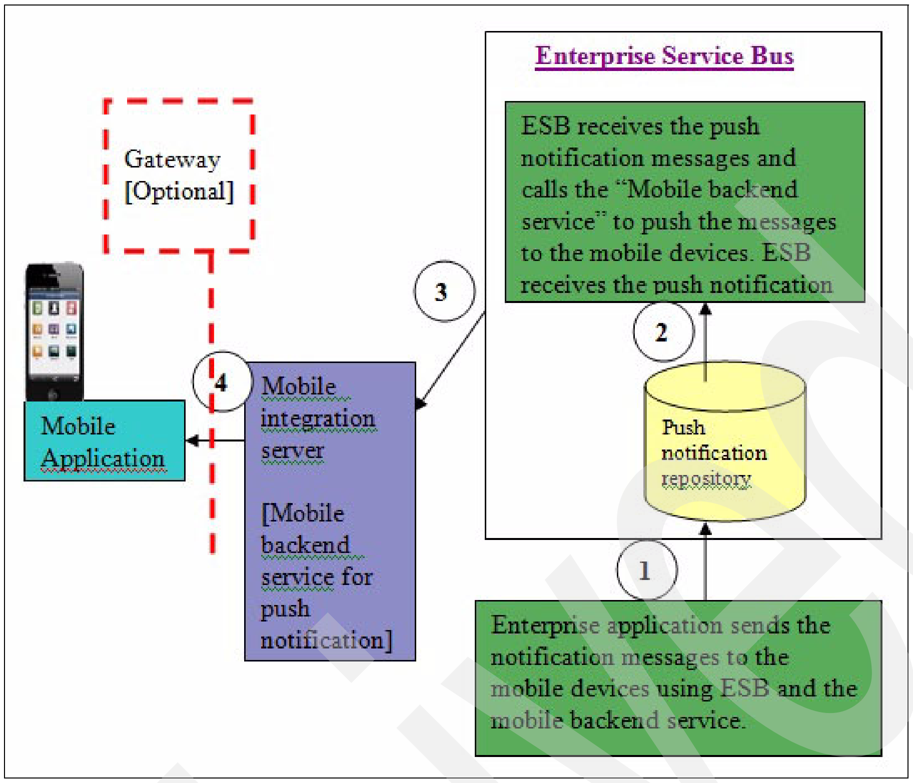

Enterprise Service Bus
ESB receives the push notification messages and calls the "Mobile backend service" to push the messages to the mobile devices. ESB receives the push notification
Gateway [Optional]
3
2
4
Mobile integration server
[Mobile backend service for push notification]
Mobile Application
Push notification repository
1
Enterprise application sends the notification messages to the mobile devices using ESB and the mobile backend service.

Figure 7-3 Push notification: Mobile Service Integration pattern

7.1.3 Level of operational control and insight

The operational control and the insight views are important parts of a system. The operational control allows you to dynamically change the system behavior when necessary, such as dynamically enabling throttle control of message processing by using a policy. The solution needs to allow monitoring of various operations and resources that are running in the system. In case of failure (such as the message rate crossing the threshold message rate due to heavy message processing through the system), an alert must be sent to the system administrator for action.

7.1.4 When to use these patterns

The Mobile Integration pattern is when an enterprise wants to externalize their services to the external world by using mobile channels. Other patterns described in this book are used for specific business operations for different applications channels.

The Mobile Integration and Push Notification patterns are applicable when enterprises in retail, banking, transport, healthcare, travel, and so on need to connect the people through mobile interfaces. The mobile interface services are built by using these patterns, so they are fast and dynamic. Several new interfaces can be added or removed easily and dynamically.

These patterns can be used to integrate heterogeneous back-end environments of the enterprise with mobile applications. They help extend systems and applications that are running in the enterprise to the mobile user. The messages are sent in synchronous and asynchronous fashion, depending on whether the mobile application is active or not active.

Archived

The following two patterns open a mobile channel for an enterprise to extend their business and services:

- Request-response pattern:

  This is used when there are *two-way* interactions between the mobile application user and an enterprise. For example, a request-response interaction is involved when a mobile user wants to do one of these tasks:

  – Pay bills
  – Make an appointment at a medical clinic
  – Buy an item from a retail shop that offers a mobile application
  – Book travel reservations by using a company mobile application that uses an asynchronous request-response for the request and confirmation

- Push Notification pattern:

  This pattern is used when there is *one-way* integration, such as when the mobile user subscribes to a service provided by the enterprise. For example, the bank pushes notification messages to their customers for any transaction in their account and the accounts' current balances. A *transaction* can be salary deposit credit or a debit for bill payment, for example. The bank can push the information to the mobile device every time that the balance changes. Mobile users can unsubscribe from these services at any time.

7.1.5 A mobile banking example

A mobile banking product can be personal banking, loan offers, credit card offers, and so on. The services can be account balance inquiry, fund transfer, loan payment, credit card payment, loan application and processing, and so on. These products and services are provided by the core banking system. Integration is required between the core banking system that is running in the bank and the mobile application. The mobile-to-core banking system integration must be highly secure and provide the required quality of service.

Mobile device users log in to their bank accounts by using a mobile application that is provided by the bank. They initiate account balance inquiry and credit card or other bill payment by using the services that are provided in the mobile application. This is a typical example, and mobile users want such services to be provided by the bank anywhere, anytime, but the benefit is mutual. Banks extend their businesses and mobile users get the ease of using the services provided by the bank.

7.2 Ways to implement these patterns with IBM products

The implementation of the Mobile Integration pattern is simple, easy, and fast when you use IBM products. The developer does not need to write application code or complex logic to implement the pattern. IBM products provide simple steps to build mobile services to integrate with enterprise back-end applications or services.

IBM Worklight platform is used to develop the mobile applications and related tasks, and the IBM Integration Bus is used to create the enterprise services and the Worklight integration adapter to connect the mobile application to the enterprise. The required business logic is written in enterprise services that are created in the IBM Integration Bus.

Archived

There are several implementation factors that must be considered when creating a mobile integration using the middleware and mobile application development platform:

- Easy to build. The solution should be easy to build by using automated steps.
- Easy to configure. The implemented solution should be easy to configure and change.
- Easy to extend. It should allow new services to be added dynamically.
- Easy to maintain. The solution should be easy to maintain by provide the flexible deployment model.
- Transactional. The requests that are sent by the mobile device must be transactional. In case of any failure, the transaction must be able to roll back.
- Throughput. The integration solution should give the required high throughput.
- Availability. The solution must be highly available.
- Security. The transactions must be secure and integrity of the data must be maintained.
- Monitoring. It is preferable to have operational monitoring of the solution to check the overall performance and to record transactional event data in a repository.

IBM Integration Bus and IBM Worklight software support all of these requirements.

7.2.1 IBM Worklight mobile application development platform

IBM Worklight software provides an advanced mobile application development platform and tools to develop software for smartphones and tablets. Worklight software helps organizations of all sizes develop, run, and manage HTML5, hybrid, and native applications by using a powerful and flexible mobile integrated development environment (IDE), next-generation mobile middleware, end-to-end security, and integrated management and analytics. It helps businesses reduce the code to shorten time-to-market for their products, and it reduces complex implementation and maintenance.

Worklight has five components:

- Worklight Studio provides a comprehensive environment for advanced, rich, cross-platform mobile application development.
- Worklight Server is mobile-optimized middleware that serves as a gateway between applications, back-end systems, and cloud-based services.
- Worklight device runtime components offer runtime client application programming interfaces (APIs) that enhance security, governance, and usability.
- Worklight Application Center helps you set up an enterprise application store that manages the distribution of production-ready mobile applications.
- Worklight console is an administrative graphical user interface (GUI) for the server, adapters, applications, and push services to help you manage, monitor, and instrument mobile applications.

Worklight supports mobile application development for multiple mobile platforms, including Android, BlackBerry, and Apple iOS, with a single code base that is compiled on the respective platforms.

In the Mobile Integration pattern that is shown in Figure 7-1 on page 64 and in Figure 7-2 on page 65, the Worklight adapter is the mobile integration adapter. The Worklight adapter runs on the server and connects to mobile applications.

Archived

Worklight adapters are the server-side code of applications that are deployed on and serviced by the IBM Worklight mobile application development platform. The adapters connect to the enterprise services that are running in the IBM Integration Bus and deliver data to and from mobile applications and perform any necessary server-side business logic on this data.

7.2.2 IBM Integration Bus software

The IBM Integration Bus is an enterprise service bus that provides universal connectivity for service-oriented environments and non-service oriented environments. It connects to a range of different systems and endpoints. It also does the universal transformation and routing of messages to integrate various applications, services, and protocols.

An integration solution is easy to develop, deploy, and manage by using the Integration Bus. It is independent of any programming language, so developers can write business logic in the programming languages of their choice, including Java, Microsoft .NET, PHP, ESQL, and so on. It allows non-programmers to do the necessary message transformation by using the Mapping Editor.

The Integration Bus supports the following industry-specific formats and protocols:

- ACORD AL3
- CSV
- EDIFACT
- FIX
- HL7
- SWIFT
- TLOG
- X12

The IBM Integration Bus Hypervisor Edition optimizes the Integration Bus for virtual environments. A range of heterogeneous platforms and databases are supported to connect to the back-end systems.

The Integration Bus reacts quickly to the changing technical and business requirements by using rule and policy-based configuration. IBM Integration Bus is also optimized for a high performance throughput and linear scaling, so your mobile solution can scale easily by adding more processor and memory capacity.

You can gain dynamic operational control by using policy-based workload management. The Integration Bus has a web-based advanced monitoring tool that provides the message processing details for flow and integration nodes, the processor and memory activities, resource use activities, and policy-based workload management by defining message rate control and threshold alert.

Figure 7-1 on page 64 and Figure 7-2 on page 65 show how you can use the IBM Integration Bus to create enterprise services in the enterprise service bus.

The IBM Integration Bus provides four built-in Mobile Integration patterns to accelerate development:

- Microsoft .NET request-response
- Mobile Service
- Push Notification from IBM WebSphere MQ
- Resource Handler

Archived

These patterns are used to implement various mobile integration scenarios. For more information, see the “Built-in patterns” section of the IBM Integration Bus Information Center:

http://bit.ly/1lTPMhj

You can also use the IBM Integration Bus to create customized Mobile Integration patterns.

7.3 Implementing the mobile banking example

The scenario: A bank wants offer bill payment and account balance inquiry services to their customers, but opening an account or loan applications or processing will be reserved for the bank staff to handle. However, the bank staff sometimes goes to the customer’s location to open an account or processes loan applications by using mobile devices. So getting these services on mobile applications ensures these services are available to the end users quickly, and processing those service requests is fast. A typical architecture to implement such a mobile banking service is shown in Figure 7-4.

An optional external gateway between the mobile application and the IBM Worklight adapter helps process high volumes of low-value traffic between the mobile application and the enterprise. Gateway security can be applied between the mobile application and the Worklight adapter.

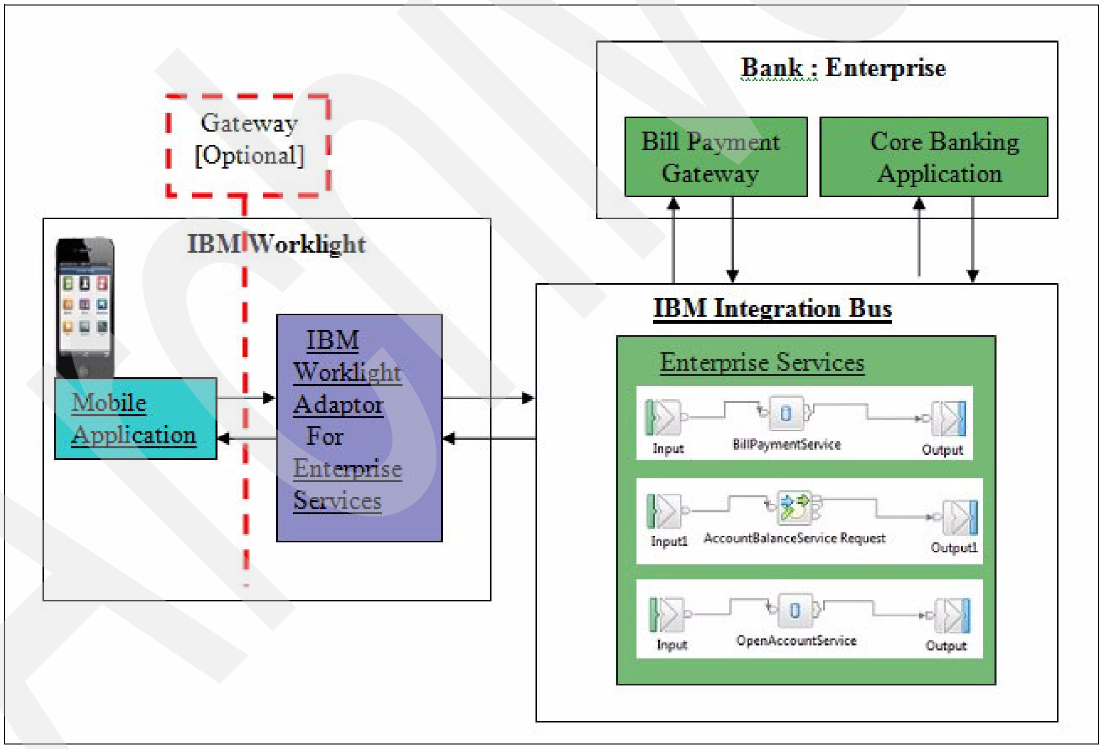


Figure 7-4 Mobile banking pattern architecture

To implement the Mobile Integration pattern in IBM Integration Bus, a new integration service can be created in the Integration Bus, and different enterprise service operations can be added to connect to the back-end systems. Several integration nodes are provided in the palette to connect to different back-end applications, databases, and services. Alternatively, a new integration service can also be created by using Web Services Description Language (WSDL).

The Mobile Integration service pattern is created by just three simple steps in the IBM Integration Bus:

1. Create a new integration service (called MobileIntegrationService in this example). This creates the enterprise service to connect to the mobile applications (see Figure 7-5).

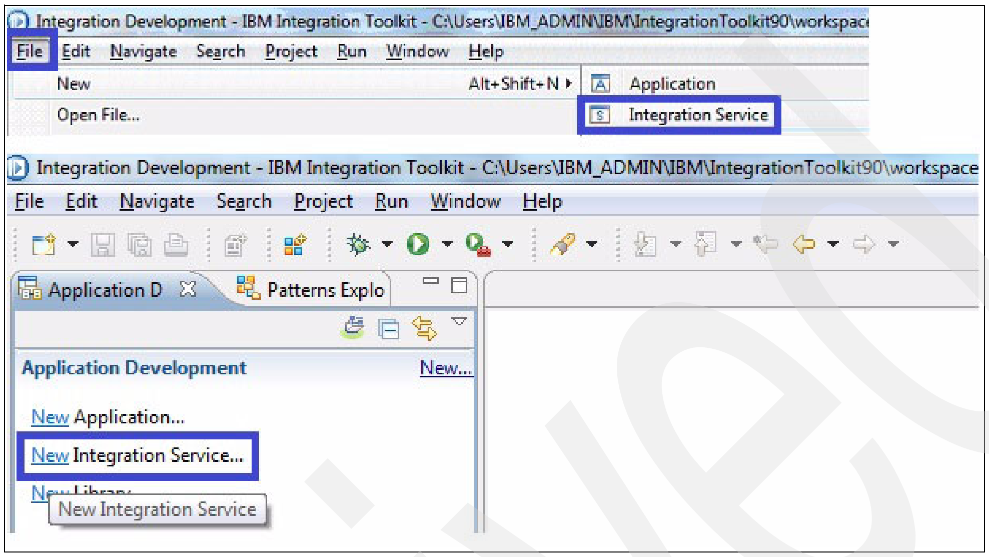

Integration Development - IBM Integration Toolkit - C:\Users\IBM_ADMIN\IBM\IntegrationToolkit90\workspace
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Application
Integration Service
Application D
Patterns Explo
Application Development
New...
New Application...
New Integration Service...
New Library
New Integration Service

Figure 7-5 Create a new integration service in the IBM Integration Bus toolkit

Archived

2. Implement the business logic in the enterprise to connect to the back-end system and get the response for the service operation. You can add multiple operations for different enterprise services (see Figure 7-6).

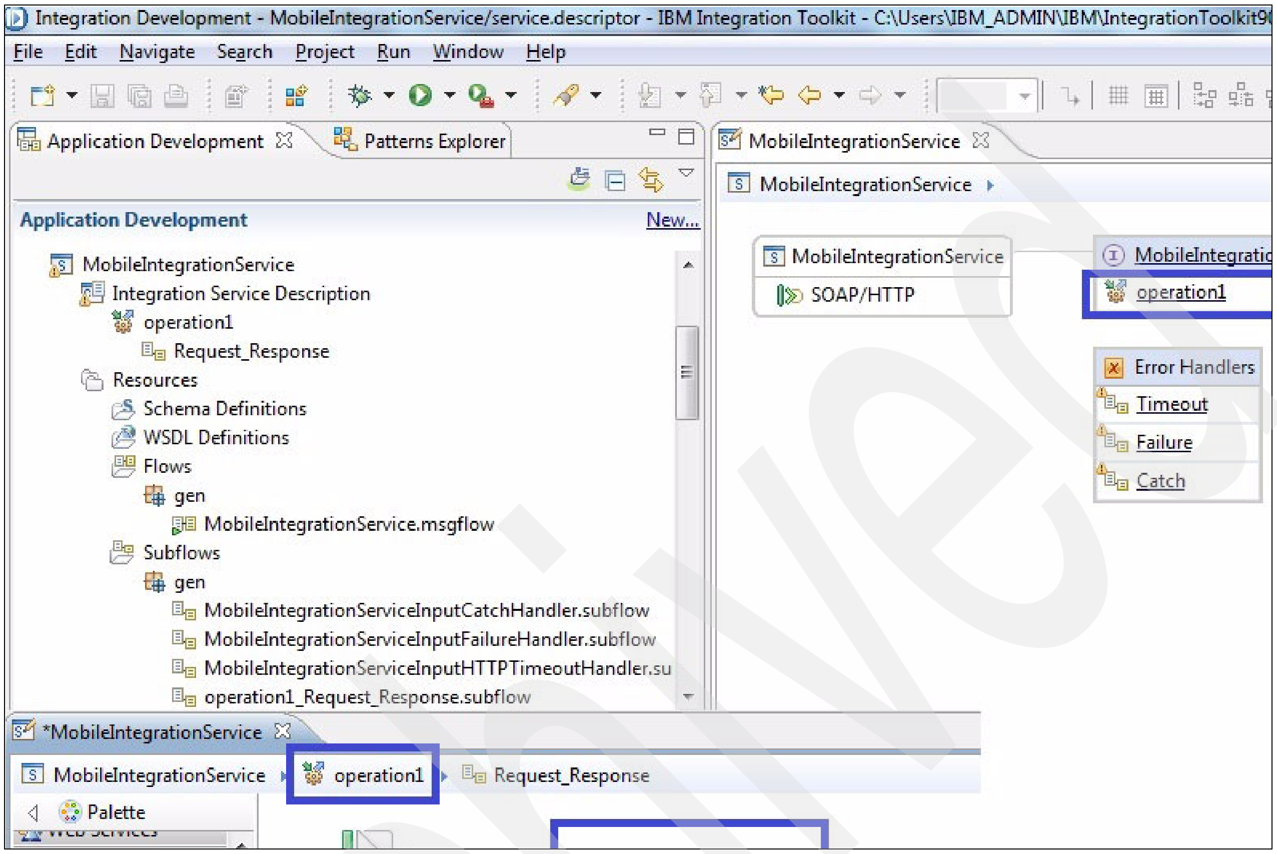

Integration Development - MobileIntegrationService/service.descriptor - IBM Integration Toolkit - C:\Users\IBM_ADMIN\IBM\IntegrationToolkit9
File Edit Navigate Search Project Run Window Help
Application Development
Patterns Explorer
Application Development
New...
MobileIntegrationService
Integration Service Description
operation1
Request_Response
Resources
Schema Definitions
WSDL Definitions
Flows
gen
MobileIntegrationService.msgflow
Subflows
gen
MobileIntegrationServiceInputCatchHandler.subflow
MobileIntegrationServiceInputFailureHandler.subflow
MobileIntegrationServiceInputHTTPTimeoutHandler.su
operation1_Request_Response.subflow
MobileIntegrationService
MobileIntegrationService
MobileIntegrationService
SOAP/HTTP
MobileIntegratio
operation1
Error Handlers
Timeout
Failure
Catch
*MobileIntegrationService
MobileIntegrationService
operation1
Request_Response
Palette

Figure 7-6 Customize the service interface to connect to the database in IBM Integration Bus toolkit

Archived

3. Using the steps that are shown in Figure 7-7 and explained in Substep a, create a mobile service pattern for the IBM Integration Bus integration service that you implemented in Step 2.

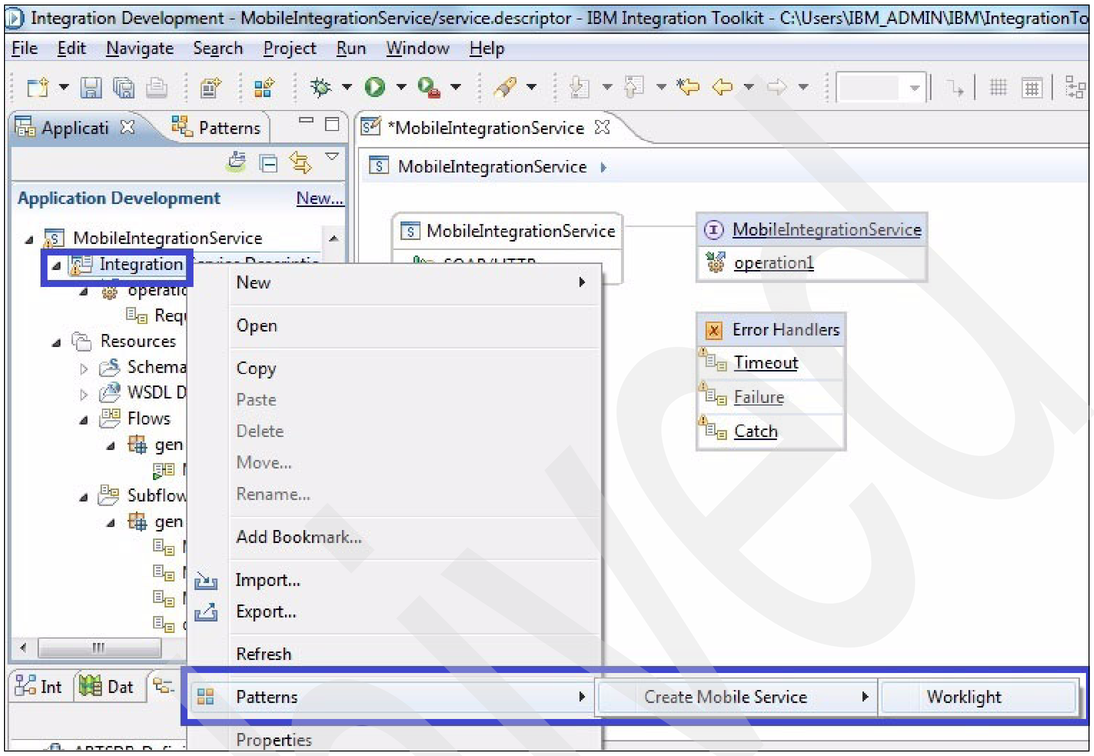


Figure 7-7 Create a mobile service Worklight pattern in the integration service in toolkit

a. Under **Integration Service Project**, select **Integration Service**, and right-click to open the drop-down menu.

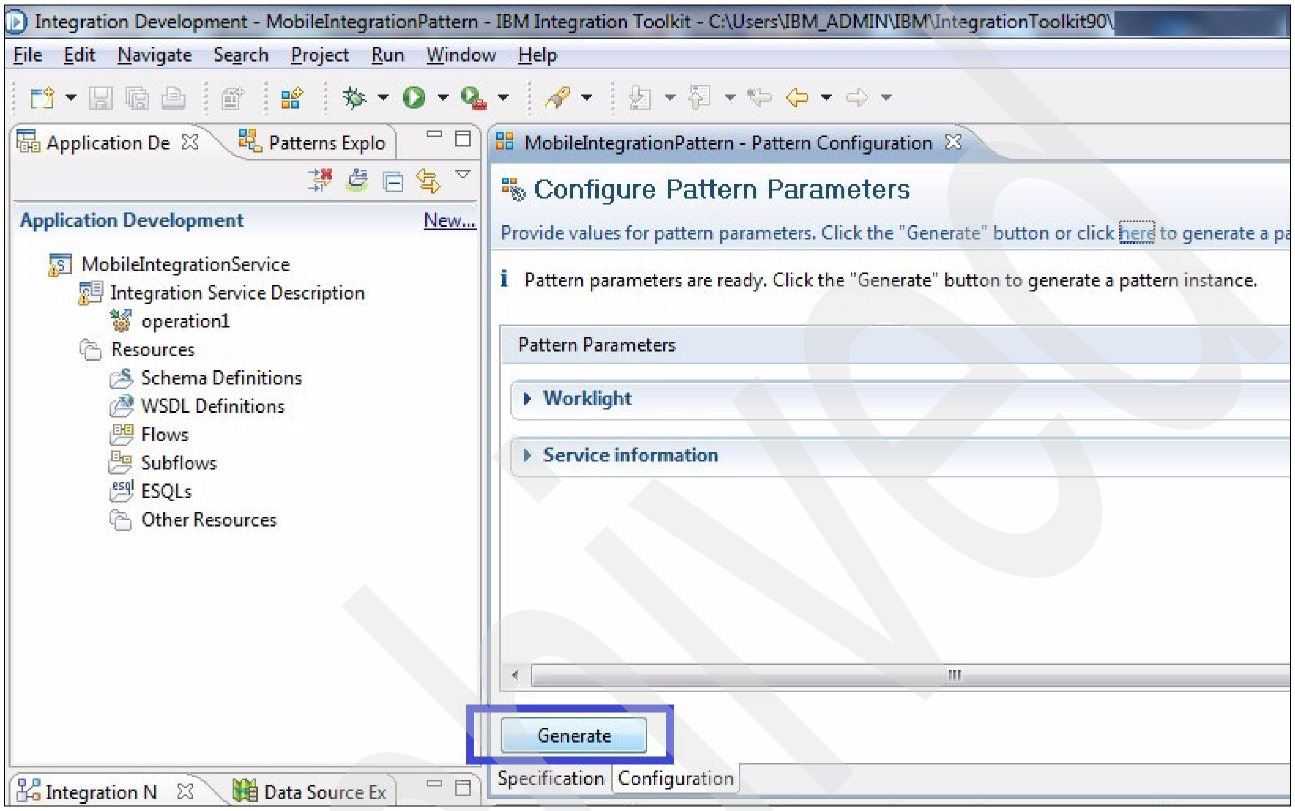
   i. Select Patterns → Create Mobile Service → Worklight.

   ii. Provide the pattern name as MobileIntegrationPattern.

Archived

This creates a Worklight pattern for the Integration Service project, as shown in Figure 7-8.

b. Select **Generate**, as shown in Figure 7-8, to generate the Mobile Integration adapter project in the IBM Integration Bus toolkit.


Integration Development - MobileIntegrationPattern - IBM Integration Toolkit - C:\Users\IBM_ADMIN\IBM\IntegrationToolkit90\
File Edit Navigate Search Project Run Window Help
Application De
Patterns Explo
Application Development
New...
MobileIntegrationService
Integration Service Description
operation1
Resources
Schema Definitions
WSDL Definitions
Flows
Subflows
ESQLs
Other Resources
MobileIntegrationPattern - Pattern Configuration
Configure Pattern Parameters
Pattern Parameters
Worklight
Service information
Generate
Specification
Configuration
Integration N
Data Source Ex

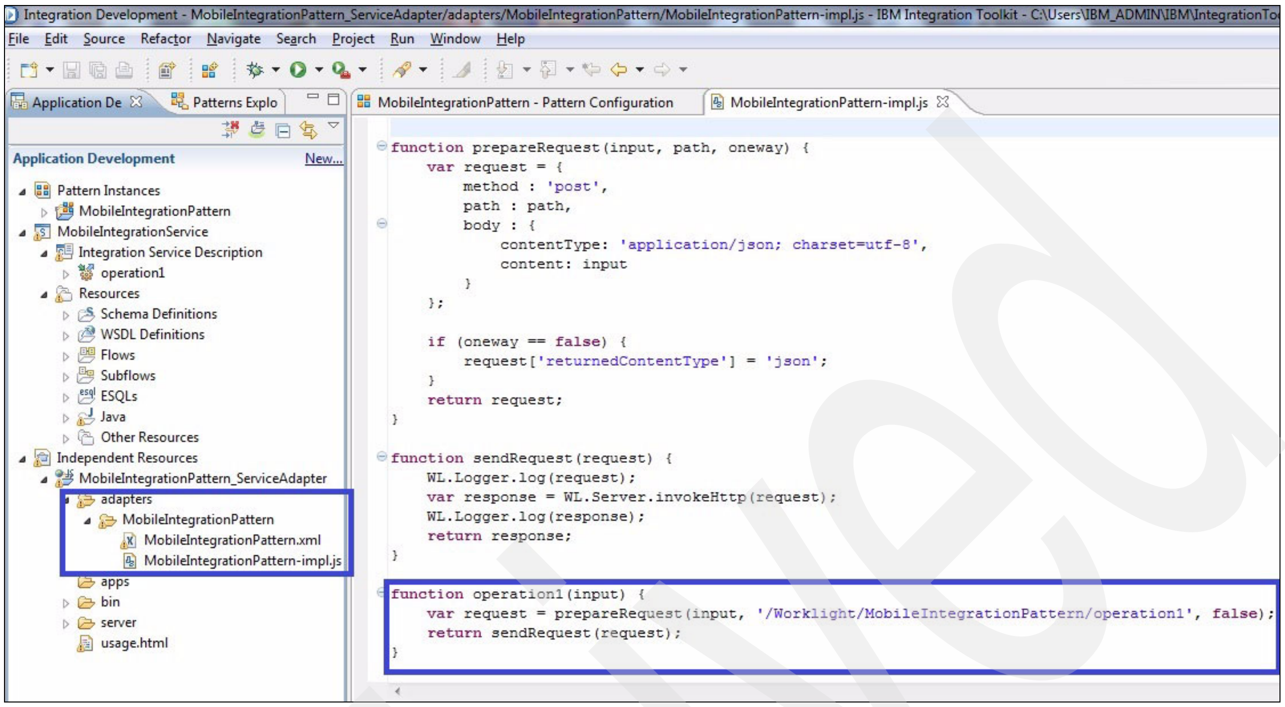
Provide values for pattern parameters. Click the "Generate" button or click here to generate a pa

Pattern parameters are ready. Click the "Generate" button to generate a pattern instance.

Figure 7-8 Mobile service Worklight pattern in Integration Service in lib

Archived

The project is used to integrate the mobile application with the ESB service operation1. The Worklight adapter project is shown in Figure 7-9.


Integration Development - MobileIntegrationPattern_ServiceAdapter/adapters/MobileIntegrationPattern/MobileIntegrationPattern-impl.js - IBM Integration Toolkit - C:\Users\IBM_ADMIN\IBM\IntegrationTo
File Edit Source Refactor Navigate Search Project Run Window Help
Application De
Patterns Explo
MobileIntegrationPattern - Pattern Configuration
MobileIntegrationPattern-impl.js
Application Development
New...
Pattern Instances
MobileIntegrationPattern
MobileIntegrationService
Integration Service Description
operation1
Resources
Schema Definitions
WSDL Definitions
Flows
Subflows
ESQLs
Java
Other Resources
Independent Resources
MobileIntegrationPattern_ServiceAdapter
adapters
MobileIntegrationPattern
MobileIntegrationPattern.xml
MobileIntegrationPattern-impl.js
apps
bin
server
usage.html

```
function prepareRequest(input, path, oneway) {
    var request = {
        method : 'post',
        path : path,
        body : {
            contentType: 'application/json; charset=utf-8',
            content: input
        }
    };

    if (oneway == false) {
        request['returnedContentType'] = 'json';
    }
    return request;
}

function sendRequest(request) {
    WL.Logger.log(request);
    var response = WL.Server.invokeHttp(request);
    WL.Logger.log(response);
    return response;
}

function operation1(input) {
    var request = prepareRequest(input, '/Worklight/MobileIntegrationPattern/operation1', false);
    return sendRequest(request);
}
```

Figure 7-9 Adapter project that is generated by the Worklight pattern in the toolkit

c. Export the **MobileIntegrationPattern_ServiceAdaptor** project from the IBM Integration Bus and import it into IBM Worklight Design Studio.

d. Deploy the Worklight adapter MobileIntegrationPattern on Worklight Server.

The mobile integration adapter is ready to use. You can call the Worklight adapter function, operation1, in your mobile application, which eventually calls the ESB enterprise services operation1.

For more information, see these web pages:

- IBM Worklight: Build, test, run and manage mobile apps:

  http://www.ibm.com/software/mobile-solutions/worklight/

- Getting started with IBM Worklight:

  http://www.ibm.com/developerworks/mobile/worklight/getting-started.html

For more information about creating a Push Notification pattern, download this presentation:

- Using Worklight API for push notifications in native iOS applications:

  http://bit.ly/PVqpyp

Archived

Archived

8

API Management pattern

This chapter describes the Application Programming Interface (API) Management pattern, which is the most recent of the integration patterns that are addressed in this IBM Redbooks publication. The API Management pattern uses APIs to integrate applications with enterprise systems and external cloud-based services.

This chapters covers the following topics:

- The value of an application programming interface
- Characteristics and capabilities of the pattern
- Ways to implement the pattern with IBM products

Archived

8.1 The value of an application programming interface

An AP is a way for services and products to communicate with each other through a documented interface. This pattern allows companies to open up data to external third-party developers, to business partners and internal departments within their company.

In the 1990's, when the World Wide Web was still new, many companies focused their business toward creating a web presence. As Internet access became more readily available, speed limitations were lifted, and technology improved, many companies that are migrated from a relatively flat and static web presence to a more dynamic, content-rich, interactive approach. Today, we live in a data-centric world of connected devices where we expect data to be readily available at our fingertips. These devices include, but are not limited to, smartphones, tablets, games consoles, and even cars and refrigerators. As the number of devices has increased, so has the complexity to manage and maintain the code for each of these devices. This is where an "API first" approach has gained the most traction. If the data is made available through a common API, this allows a single point of maintenance, security, versioning, and control. In this way, data can be available consistently and across multiple devices. Also, by making APIs available and sharing them through social media, companies can outsource application development to third parties or business partners, which is becoming increasingly important as the number of devices increase.

APIs can help companies make data available to the outside world or to selected business partners. These APIs can be used to create applications to market a company's products and develop new markets and opportunities. After APIs are established, they can be used to develop brand awareness and increase profit. Most importantly the APIs, which are a now a core part of the business also need to be treated as a product. Whether you or your company are considering using APIs, it is likely one of your competitors are. In our highly competitive world, this is a significant reason to consider adopting an API strategy.

8.2 Characteristics and capabilities of the pattern

As we have established, it is important to treat an API as one of your products. Although you can make APIs available by using the Gateway pattern, as discussed in Chapter 7, "Mobile Integration pattern" on page 63, the API Management pattern is a superset of the Gateway pattern and builds upon it by providing more functionality to help you do this. To treat your API as a product, you need an API strategy, a business model, a way of socializing the API, information about the API use, and so on. To fully address these concerns, the API Management pattern must meet the requirements of the following roles:

- **API Product Manager:**

  The API product manager establishes the API strategy and pricing models. They are responsible for the marketing and socialization of the APIs and use the available analytics to evaluate the existing APIs and make investment decisions.

- **API Developer:**

  The API developer is responsible for creating, testing, versioning, applying security and publishing the APIs.

- **Application Developer:**

  The application developer subscribes to the publicized APIs, which they use to create applications from, such as a smartphone application, a web page etc.

- **Operations Lead:**

  The operations lead is responsible for the initial installation and configuration of the API Management portal, as well as the day-to-day running and maintenance of the environment. In a software as a service (SaaS) cloud solution, it is worth noting that this role is outsourced and included as part of the hosting cost.

8.2.1 Business pricing models for APIs

There are several common business models for API exposure. Which model is appropriate for a particular API depends on the goals of the API provider, which might include monetization, brand awareness, and so on. These include but are not limited to free APIs, developer pays, developer gets paid, and indirect models:

- **Free APIs:**

  Available for use at no charge, such as the APIs for popular social networks. Free APIs can drive adoption of APIs and brand loyalty and help the API provider enter new marketing channels. Examples of free APIs are limited. Although there is usually no direct business benefit gained from making free APIs available, most of the examples have an indirect business benefit (see what follows).

- **Developer pays:**

  In most cases, developers pay for use to help their own business ventures, as a cost of doing business. There are also significant subcategories within this business model:

  – **Pay as you go.** Pricing is determined by metered use. For example, the price for using a cloud computing might be determined by the operating system and the size of the solution on an hourly basis.

  – **Tiered.** Developers pay for use of a particular tier, based on the number API calls over a fixed time period. Although the cost increases per tier, the cost per API call usually drops. Vertical resources use the tiered business model. Prices drop with using more volume (API calls), so after analyzing their use over time, users can adjust their tiers. This works similarly to how cell phone voice plans have worked for the past 10 years, where levels of phone calls determine the per-call charge.

  – **Use-based.** Pricing is determined by use and based on units of measure, such as API calls. For example, a cloud storage company might charge per gigabyte of use per month.

  – **Transaction fee:** Pricing is dependent upon the value of the transaction. An electronic money transfer company might charge a percentage of the transaction as their fee for providing payment services.

  – **Freemium.** Companies offer developers some of their API capabilities for no charge and then charge for additional functions. For example, a web mapping service might allow a low number of calls to be made to the API for free and charge for any additional calls to the API.

- **Developer gets paid:**

  In this model, the developer receives payments for extending the business model of others. There are two major categories for this model:

  – **Revenue sharing.** Developers receive a small percentage of total revenue, based on a business agreement with the provider. For example, a mobile application store might charge developers a yearly subscription, and the developer receives a percentage of sales revenue for their applications.

– **Affiliate.** Developers receive increasing revenue (percentage-based, fixed price, or tiered scale), based on extending partner solutions. For example, a shopping price-comparison service might pay developers a small portion of revenue that is based on the volume of revenue that their APIs generate.

- **Indirect:**

  This business model does not include direct monetary exchange, but the API providers benefit significantly from making these types of APIs available by generating revenue in other ways. For example, a streaming media company can use an API to make it easier for many consumer devices to easily support them, which helps increase the number of subscribers. An online auction company could offer an API to give developers access to trading services, which extends the reach of listings and helps increase revenue.

8.2.2 Components of an API Management pattern

The API Management pattern consists of the API Management portal, the API user, and the enterprise services that are made available. These enterprise services might or might not be made available through an enterprise service bus (ESB), see Figure 8-1.

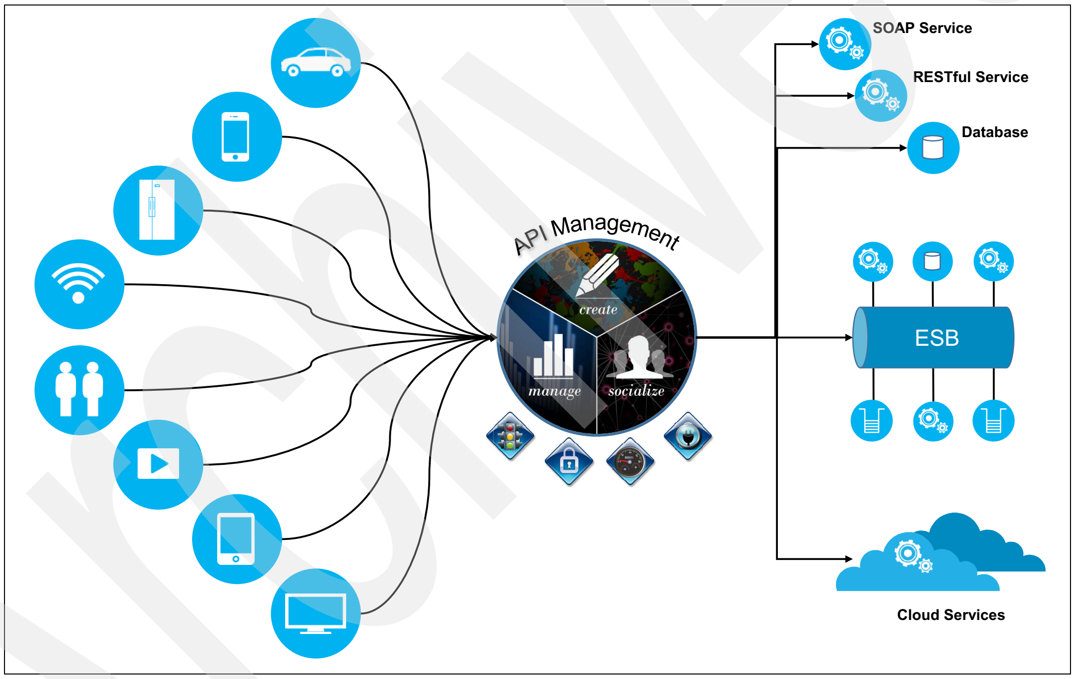


Figure 8-1 Components of the API Management pattern

Each API consumer sends a request to the API Management portal, which interacts with the enterprise service before sending a response to the consumer. APIs use industry-standard message formats and protocols, such as SOAP for web services and XML or JavaScript Object Notation (JSON) for Representational State Transfer (REST). The API management solution typically acts as a simple proxy that forwards the request and response messages from the back end. In advanced cases, it might also aggregate or mash up responses from multiple systems or perform transformations if the back-end systems use a different message format or protocol.

ARCHIVED

8.2.3 Level of operational control and insight

As illustrated in Part 1, "Topology of an integrated enterprise", important aspects of building composable business systems are policy-based operational control and the ability to collect insight through sensing of operational interactions. The Gateway pattern already provides a high degree of IT operational control.

What the API Management pattern adds is a set of business controls and related business insight. For example, from an IT operations perspective, the policy authors are primarily concerned with the amount of traffic that an API is capable of handling, so the authors create policies to protect the services that they are making available. However, from the business perspective, the policy authors are focused on the contracts between the API consumer and provider and enforcing and monitoring such agreements.

Good implementations of the API Management patterns supports both types of operational control and insight: The business product aspect that is driven by the business owner of the API and the IT operational aspects that are driven by the person responsible for the operational health of the system.

8.2.4 When to use the API Management pattern

There are four typical adoption patterns for the API Management pattern: Internal, Mobile, Partner, and External.

- **Internal:**

  API Management adopters want to easily control and manage the API consumption within the enterprise (SOAP, REST). Typically, this is where different departments or lines of business want to "socialize" their APIs by sharing them on developer portals for other lines of business, monitor who is using their APIs, and add a level of security and runtime control. In addition, it is desirable to use security and use policies to both simplify and add consistency across all of their APIs.

- **Mobile:**

  API Management adopters would like to make data available to be used by various mobile devices, such as mobile phones, tablets, utility meters, refrigerators, and thermostats. The enterprise often does not have the resources to do mobile development internally, so they need to secure and control access to data for third-party development. In this case, they require consistent reports of consumer data across all channels (mobile, web, traditional applications), which can require them to re-evaluate their digital strategy to put APIs at the foundation. This scenario is often associated with external consumers. However, companies are often required to make their internal data available to their mobile workforce in a secure, easy, and efficient manner. IBM is an example of a company that does this.

- **Partner:**

  API Management adopters want a faster, consistent way to connect with multiple partners. In some cases, the partners are forcing them to produce the API.

- **External:**

  API Management adopters are seeking to disrupt existing value chains, increase brand loyalty, and open new channels. They are taking an API first approach to new products.

Archived

8.2.5 Retail example

A leading electronics retailer has a high cost of sale because of their reliance on retail stores. They have no participation in external, web-based, or mobile markets and limited access to customers in locations without storefronts.

They solved this problem by publishing APIs that target content aggregation providers, such as price comparison websites and product review websites. The APIs show their product catalog, price list, and inventory by store locations. They increase revenue through participation in external markets, increase profitability by closing under-performing stores, and differentiate their products in external markets, such as Amazon.com, by providing same-day fulfilment through web storefronts.

8.3 Ways to implement the pattern with IBM products

The API Management portal is an enhanced gateway that provides the features summarized in the “Marketplaces” and “Capabilities for Service Providers” sections of the IBM reference architecture for API Management, as shown in Figure 8-2.

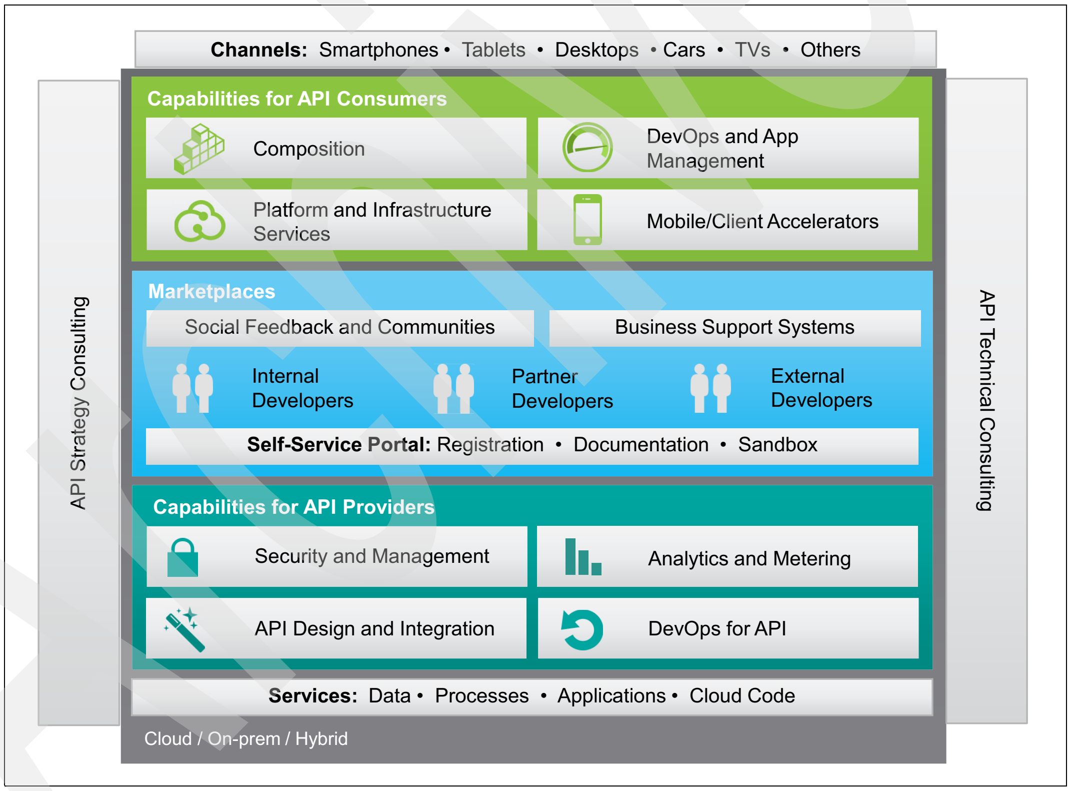

Channels: Smartphones • Tablets • Desktops • Cars • TVs • Others
Capabilities for API Consumers
Composition
DevOps and App Management
Platform and Infrastructure Services
Mobile/Client Accelerators
Marketplaces
Social Feedback and Communities
Business Support Systems
Internal Developers
Partner Developers
External Developers
Self-Service Portal: Registration • Documentation • Sandbox
Capabilities for API Providers
Security and Management
Analytics and Metering
API Design and Integration
DevOps for API
Services: Data • Processes • Applications • Cloud Code
Cloud / On-prem / Hybrid
API Strategy Consulting
API Technical Consulting

Figure 8-2 IBM reference architecture for API management

In the “Marketplaces” section, the API Management software provides a developer portal. The portal should allow self-service registration and manage unique identifiers for applications. Application developers should be able to browse a registry of available APIs, read documentation to learn how to use the APIs, and use test tools to try the APIs.

In the “Capabilities for API Providers” section, the API Management portal should support the API development lifecycle. API developers use it to create, secure, deploy, test, debug, and version APIs and monitor their use. The API product manager uses the analytics to evaluate the performance of APIs and, potentially, bill developers for use.

At run time, the API Management portal should also perform the usual gateway tasks such as policy enforcement, caching, routing, transformation, and monitoring. The Operations Lead can monitor the health of the API Management portal and the environment in which it is running. The Ops Lead will also be able to scale the environment to meet increased API traffic and ensure high availability.

8.3.1 IBM API Management benefits

IBM API Management is a unified on-premises environment that provides an intuitive user experience for managing the complete API lifecycle, from creation, publication, and adoption, to support and monitoring. It provides companies with the tools for creating, using proxies, assembling, securing, scaling, and making APIs available to developers through a customizable developer portal. This software enables businesses to attract and engage with application developers to foster increased use of the published APIs. The API Management robust administration portal allows companies to easily establish policies for critical API attributes, such as self-registration, quotas, key management, and security policies. The robust analytics engine provides valuable role-based insight for both API owners, solution administrators, and application developers.

API Management enables companies to realize the maximum value from their APIs by providing the following key features and benefits:

- Secures, scales, and controls access to APIs to provide a resilient and flexible API runtime infrastructure. API Management is powered by IBM WebSphere DataPower Gateway appliances.
- Empowers companies with the insight to change and grow their business in the new API economy with robust business analytics.
- Rapidly addresses business demands with the creation of new APIs from existing business assets or simple configuration-based cloud services integration.
- Nurtures innovation by building a community that attracts developers, entrepreneurs, and partners who will rapidly build new applications and extend the value of your core enterprise assets.

8.3.2 Mapping an IBM API Management portal to the API Management pattern

The IBM API Management environment meets the requirements of all of the following roles in a single solution:

- **API product manager:**

  The API product manager can manage APIs with business control. The manager can define entitlements to specify the number of API calls a specific application is allowed to make, approve or reject requests for entitlements, contact application developers through email, or block malicious API users from making further API calls. The manager can also quickly define a branded developer portal to share APIs with the developer community, easily generate documentation, and control the visibility of APIs. And the manager can view business analytics for the APIs, search and filter analytics data, and export use data for billing purposes.

- **API developers:**

  API developers can iteratively develop, test, version, and deploy APIs, all from a single interface. The developers can quickly assemble new APIs through a configuration-not coding approach. Developers can secure and scale APIs with WebSphere DataPower technology, which provides built-in caching, quota management, and flood control.

- **Application developers:**

  Application developers can register their applications, browse API documentation, subscribe, and request entitlement for an API and test the API, all from a single self-service portal. This gives application developers access to APIs to learn how to use them in minutes.

- **Operations leads:**

  The operations leads can view monitoring information and logs for the environment. The ops leads can react to current use data and scale the environment elastically by adding and removing appliances in a few simple clicks. They can partition APIs for different departments or stages of development into separate tenants. They can also have full control of the environment and data with on-premises deployment.

8.3.3 Implementing the retail example

Figure 8-3 illustrates an IBM API Management implementation of this retail example.

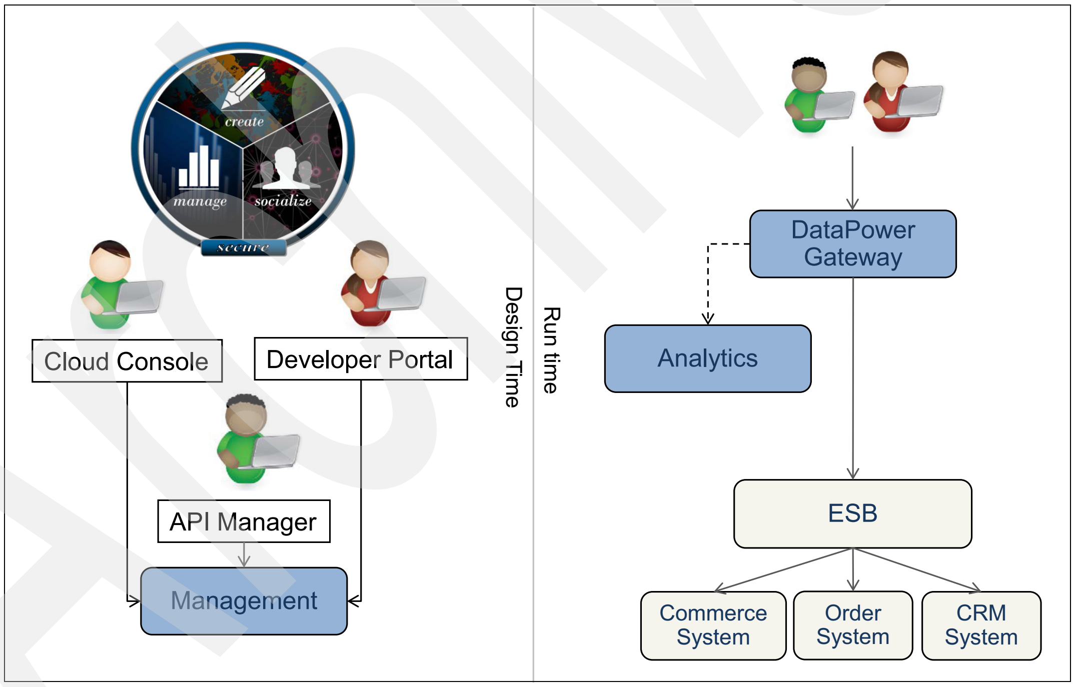


Figure 8-3 IBM API Management patter retail store implementation

The white boxes in the lower right of the diagram are the internal systems that a typical retailer uses to store their product and customer information, and to process orders. In this case, all requests to them go through the enterprise service bus.

Each IBM API Management environment consists of three logical components: management, gateway, and analytics, as shown in the blue boxes in the diagram. The management component hosts the administrator and developer portals and stores the configuration that is created with them. At design time, it can connect to internal systems to test connectivity or

Archived

query for the metadata that is required by the graphical mapping tool. At run time, all requests are received by the gateway component, which enforces entitlements and other policies before routing them to the internal systems. The gateway component also sends analytics data to the analytics component asynchronously to prevent performance degradation under heavy loads.

At design time, the operations lead uses the Cloud Console portal to set up the environment and create a tenant for the API developer to use. The API developer uses the API Manager portal to create, test, and secure the APIs to make available. The application developer uses the Developer Portal to register to use the APIs before making requests to them at run time.

Archived

Archived

Related publications

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

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *Exposing and Managing Enterprise Services with IBM API Management*, SG24-8193
- *IBM WebSphere Application Server V7.0 Web Services Guide*, SG24-7758
- *Service Lifecycle Governance with IBM WebSphere Service Registry and Repository*, SG24-7793
- *Unleashing DB2 10 for Linux, UNIX, and Windows*, SG24-8032
- *Introduction to Networking Technologies*, GG24-4338
- *WebSphere MQ Primer: An Introduction to Messaging and WebSphere MQ*, REDP-0021

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- *The Nexus of Forces: Social, Mobile, Cloud and Information*, a Gartner special report:

  http://www.gartner.com/technology/research/nexus-of-forces/
- IBM API Management

  http://www.ibm.com/software/products/en/api-management
- Systems of Interaction:

  http://www.ibm.com/software/solutions/systems-of-interaction/
- Systems of Engagement and the Enterprise:

  http://www.ibm.com/software/ebusiness/jstart/systemsofengagement/
- IBM SOA reference model, *IBM SOA Foundation: Providing what you need to get started with SOA* (2005 white paper):

  ftp://ftp.software.ibm.com/software/soa/pdf/SOA_g224-7540-00_WP_final.pdf
- Service-oriented architecture (SOA): Simply good design (includes link to download the ebook titled *SOA Design Principles for Dummies*):

  http://www.ibm.com/software/solutions/soa/

Archived

- Open Group SOA ontology:

  https://www2.opengroup.org/ogsys/catalog/C144
- CDInput node, IBM Integration Bus Information Center:

  http://pic.dhe.ibm.com/infocenter/wmbhelp/v9r0m0/topic/com.ibm.etools.mft.doc/bc14020_.htm
- CDOutput node, IBM Integration Bus Information Center:

  http://pic.dhe.ibm.com/infocenter/wmbhelp/v9r0m0/topic/com.ibm.etools.mft.doc/bc14015_.htm
- "Using WebSphere Service Registry and Repository V8 and DataPower V5 for service level mediation policy enforcement," an IBM developerWorks article:

  http://www.ibm.com/developerworks/websphere/library/techarticles/1212_willoughby/1212_willoughby.html
- XML-based attacks, an IBM WebSphere Developer Technical Journal article:

  http://www.ibm.com/developerworks/websphere/techjournal/0603_col_hines/0603_col_hines.html
- IBM Integration Bus: General industry standards supported

  http://pic.dhe.ibm.com/infocenter/wmbhelp/v9r0m0/index.jsp?topic=%2Fcom.ibm.etools.mft.doc%2Fah36080_.htm
- IBM Worklight overview:

  http://www.ibm.com/software/mobile-solutions/worklight/
- IBM Worklight, Get Started:

  http://www.ibm.com/developerworks/mobile/worklight/getting-started.html
- IBM Integration Community:

  https://www.ibm.com/developerworks/connect/integration

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Archived

Integration Throughout and Beyond the Enterprise

Change the enterprise with integration patterns

Learn the recipe for business integration

Discover the topology of an engaging enterprise

Throughout the history of the IT industry, integration has been an important part of most projects. Whether it is integration of transactions, data, or processes, each has challenges and associated patterns and antipatterns. In an age of mobile devices, social networks, and cloud services, and big data analytics, integration is more important than ever, but the scope of the challenge for IT projects has changed.

Partner APIs, social networks, physical sensors and devices, all of these and more are important sources of capability or insight. It is no longer sufficient to integrate resources under control of the enterprise, because many important resources are in the ecosystem beyond enterprise boundaries. With this as the basic tenet, we address these questions:

- What are the current integration patterns that help enterprises become and remain competitive?
- How do you choose when to use which pattern?
- What is the topology for a "composable business"?
- And how do you accelerate the process of implementation through intelligent choice of supporting integration middleware?

This IBM Redbooks publication guides integration practitioners and architects in choosing integration patterns and technologies.

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.

For more information:
ibm.com/redbooks

SG24-8188-00 ISBN 0738439630

Archived

IBM Redbooks

Integration Throughout and Beyond the Enterprise

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Archived

Archived