IBM

Exploiting IBM System z in a Service-Oriented Architecture

Modernizing application environments

Using IBM SOA Reference Architecture software

Leveraging System z quality of service

Alex Louwe Kooijmans
Nick Baguley
Simon Chan
Ignacio Perez Gonzalez
Foulques de Valence
Peter Verspecht

ibm.com/redbooks

Redbooks

International Technical Support Organization

Exploiting IBM System z in a Service-Oriented Architecture

February 2009

SG24-7651-00

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

First Edition (February 2009)

This edition applies to a large number of products, standards and solutions, but not to any version, release, or modification specifically. Refer to the respective chapters for details regarding version, release or modification level, if appropriate.

Contents

Notices

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

AIX®
CICSPlex®
CICS®
ClearCase®
Cognos®
DataPower device®
DataPower®
DataStage®
DB2 Connect™
DB2 Universal Database™
DB2®
Domino®
DS6000™
DS8000®
Enterprise Storage Server®
ESCON®
FICON®
FileNet®
FlashCopy®
GDPS®
Geographically Dispersed Parallel Sysplex™
HiperSockets™
HyperSwap™
i5/OS®
IBM®
Informix®
InfoSphere™
Lotus Notes®
Lotus®
Maximo®
NetView®
Notes®
OMEGAMON®
Optim™
OS/390®
Parallel Sysplex®
PR/SM™
pureXML®
RAA®
RACF®
Rational®
Redbooks®
Redbooks (logo) ®
RequisitePro®
System Storage™
System z10™
System z9®
System z®
Tivoli Enterprise Console®
Tivoli®
TotalStorage®
VTAM®
WebSphere®
z/OS®
z/VM®
z/VSE™
z9®
zSeries®

The following terms are trademarks of other companies:

Cognos, and the Cognos logo are trademarks or registered trademarks of Cognos Incorporated, an IBM Company, in the United States and/or other countries.

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

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

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

SAP NetWeaver, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

EJB, J2EE, Java, JavaScript, JDBC, JMX, JVM, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Preface

There are many options to implement a service-oriented architecture (SOA). A good SOA solution for one company might not be a good solution for another, even in the same business environment and IT landscape. Choosing a solid SOA solution involves strategy, vision, architectural thinking, and finally, technology. It also involves personal taste and organizational politics.

Although this book focuses in on SOA solutions on the System z® platform, we would like to remind you that the maximum benefits of an SOA solution can only be achieved if the entire IT landscape is taken into account. We do not cover SOA implementation aspects on other platforms, though, because so much has been written already about these topics.

So, why did we decide to write a book dedicated to SOA solutions on System z then? The answer is quite simple: the combination of the System z hardware, the operating systems that run on System z, and the SOA middleware provide specific functionality and influence the effectiveness of your SOA solution to a great extent. In other words, if you were to implement the same SOA solution on System z and on another platform, you would see different results.

The team that wrote this book

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

Alex Louwe Kooijmans is a project leader with the International Technical Support Organization (ITSO) in Poughkeepsie, NY, and specializes in SOA technology and solutions on System z. He also specializes in application modernization and transformation on z/OS®. Previously he worked as a Client IT Architect in the Financial Services sector with IBM® in The Netherlands, advising financial services companies on IT issues such as software and hardware strategy and on demand services. Alex has also worked at the Technical Marketing Competence Center for zSeries® and Linux® in Boeblingen, Germany, providing support to customers getting started with Java™ and WebSphere® on System z. From 1997 to 2000, Alex completed a previous assignment with the ITSO, managing various IBM Redbooks® projects and delivering workshops around the world in the areas of WebSphere, Java, and e-business technology on System z. Before 1997 Alex held a variety of positions in application design and development, product support and project management, mostly in relation to the IBM mainframe.

Nick Baguley is an Applications Architect working in the financial sector in South Africa. He has 20 years of experience on z/OS. His areas of expertise are legacy rejuvenation and service enablement.

Simon Chan is an IBM Certified Senior IT Specialist in WebSphere. He joined IBM in 1998 and is an Infrastructure Architect for Infrastructure Services, Securities Industry Services, FSS/GTS, for IBM Canada. Simon has rich experience in IBM System z infrastructure, Linux on System z infrastructure, WebSphere on z/OS and distributed systems. He is currently responsible for the SOA infrastructure enablement for Securities Industry Services.

Ignacio Perez Gonzalez is a Consulting Client IT Architect in Madrid, Spain. He has 10 years of experience in IBM, and additional experience prior to IBM as a system administrator. He holds a degree in Computer Science from the Universidad Politécnica de Madrid. He started his IBM career in Global Business Services as an Application Architect, and has worked with most IBM systems and software solutions, in particular UNIX servers and WebSphere products. Most recently he has focused on Public Sector industries solutions and SOA architecture design. He has written on topics related to mobile e-business solutions.

Foulques de Valence is a System z Web IT Architect with IBM and is currently based in Poughkeepsie, NY, where he works for the STG Lab Services team. He provides consulting services in the USA and worldwide concerning SOA, security and WebSphere products, with a focus on System z infrastructures. Previously, he was a Web infrastructure IT Architect in France specialized in SOA and z/OS. Foulques is a co-author of multiple Redbooks publications dealing with WebSphere on z/OS. He received a Master Degree in Computer Science and Engineering from Ensimag in France. He furthered his education at the State University of New York at Buffalo and at Stanford University, CA, USA.

Peter Verspecht is an IBM Certified IT Specialist working as SOA Infrastructure Architect for IBM Global Technology Services in Belgium. He joined IBM in 1998 and has 18 years of experience on z/OS, mainly in transaction (CICS® and WebSphere) and database management (DB2®). In recent years he also built skills in WebSphere and DB2 on distributed. He is an IBM Certified Administrator for SOA Solutions - WebSphere Process Server V6.0, IBM Certified SOA Solution Designer, and IBM Certified SOA Associate.

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

Dominique Richard
IT Specialist, IBM PSSC Montpellier

The team, from left to right: Simon Chan, Nick Baguley, Ignacio Perez Gonzalez, Foulques de Valence, Peter Verspecht and Alex Louwe Kooijmans

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

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 in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an e-mail to:
  redbooks@us.ibm.com
- Mail your comments to:

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

1

Introduction

From a high-level perspective, the benefits of SOA are clear to most people in the IT industry. In fact, the objective of implementing SOA can be simply summarized with three key phrases: *business-IT alignment*, *reusable services* and *loosely coupled integration*.

- *Business-IT alignment* simply means that there is a clear definition and flexible implementation of IT components supporting certain business functions. Each change in the business must be accommodated easily by the IT landscape and there is a clear mapping between each business function and the IT components supporting that function.
- *Reusable services* are IT components with an open interface and a well thought out scope. It is this combination that makes a piece of IT a service that can be reused easily.
- *Loosely coupled integration* reduces the dependencies between IT components significantly. This makes it easier to make adjustments based on business changes.

Mandatory enablers for SOA are *governance*, *organization* (including all the politic factors) and *technology*. Without these factors it becomes very difficult, even impossible, to successfully implement SOA.

1.1 Objectives of this book

This book begins with the assumption that you have already embraced SOA and understand the high-level concepts and benefits. For most people the real challenge starts when an implementation needs to be designed. Because SOA quickly impacts the entire IT landscape, one of the biggest challenges becomes finding a starting point. Furthermore, defining the way to transition within a reasonable time frame and budget is probably even more complex than designing the solution.

The IBM System z is a high-end environment. Typically, high-end products are characterized by the fact that you can obtain extreme value with such a product, but only if you have deep knowledge of how to deal with the technology. This is the reason why System z in many cases needs dedicated publications. This is also the case for implementing SOA on System z. And in order to successfully implement an SOA solution on System z you need to architect and design the solution first.

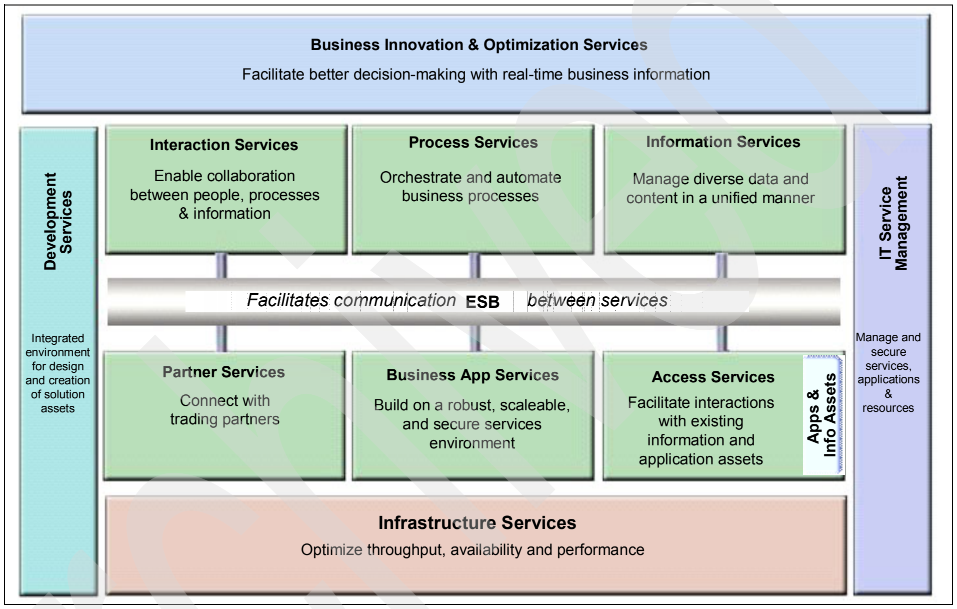
In this book we address most of the typical questions and design discussions that come up when designing an SOA solution on System z. One of the first questions a company might consider is whether System z is the right platform on which to implement all or part of an SOA solution. In this book we do not focus on proving the value of the platform for SOA, and assume you have already chosen to implement at least part of an SOA solution on System z. There is a lot of SOA technology available and we provide guidance to help you make the right decisions.

1.2 How this book is organized

When we started to write this book we realized the best way to ensure that we touch on all the relevant topics would be to use the IBM SOA Reference Architecture as the structure for our chapters and discussions. We created a chapter for each area of the IBM SOA Reference Architecture. Those areas are shown in Figure 1-1 on page 3 and include:

- ESB services, discussed in Chapter 2, “Enterprise Service Bus services” on page 5. The Enterprise Service Bus is the integration layer between all the other types of services.
- Business Process services, discussed in Chapter 3, “Business Process Services” on page 101.
- Business Application services, discussed in Chapter 4, “Business Application services” on page 165.

- Interaction services, discussed in Chapter 5, “Interaction services” on page 217.
- Information services, discussed in Chapter 6, “Information services” on page 267.
- Service Management services, discussed in Chapter 7, “SOA Service Management” on page 343.
- SOA Lifecycle services, discussed in Chapter 8, “Service life cycle management” on page 459.



Business Innovation & Optimization Services
Facilitate better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enable collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manage diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
Facilitates communication ESB between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitate interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimize throughput, availability and performance

Figure 1-1 IBM SOA Reference Architecture

Each chapter has a common layout and includes at least the following topics:

- Definition, scope, and functions
- Key architectural decisions to be made with respect to System z
- Patterns
- Platform choice criteria
- Solutions, categorized in types of solutions, if necessary

1.3 The audience for this book

This book is written with the following readers in mind: IT Architects, especially solution architects, enterprise architects, infrastructure architects, and application architects.

Ideally, the reader of this book is an architect with the mission of designing an SOA solution and knowing that System z might need to be part of the solution.

2

Enterprise Service Bus services

In this chapter we zoom in on the Enterprise Sevice Bus (ESB) portion of the IBM SOA Reference Architecture, one of the most important components of an SOA. The topics covered are:

2.1 Definitions, scope, and function

In this section we introduce the ESB and describe its scope and boundaries within the IBM SOA Reference Architecture.

2.1.1 Identifying the ESB

An Enterprise Service Bus (ESB) is a pattern for an abstract layer that stands as a hub connecting different business solutions. In the SOA architecture blueprint it is referred to as the *bus* component. It can be composed of one or many software products, and also can be the result of an ad-hoc development done in a company. The ESB provides interconnectivity services that are used by business solutions that interact with each other in different ways, such as:

- Synchronous or asynchronous
- Persistent or non-persistent
- Loosely-coupled or tightly-coupled

The ESB facilitates connectivity by abstracting communication protocols, data formats, and routing logic.

The ESB provides a set of connectors used by business applications to send messages through the ESB. It also supports different kinds of communications protocols, data formats, and languages that can be used to create *mediation* logic. The most popular connector is the Web Services connector, which permits communication through Web Services protocols and standards such as SOAP, WSDL, and XML.

Important: The exact functionality, standards, languages, and so forth, provided by the ESB, depends on the product chosen.

Figure 2-1 on page 7 illustrates the role of the ESB in the IT landscape.

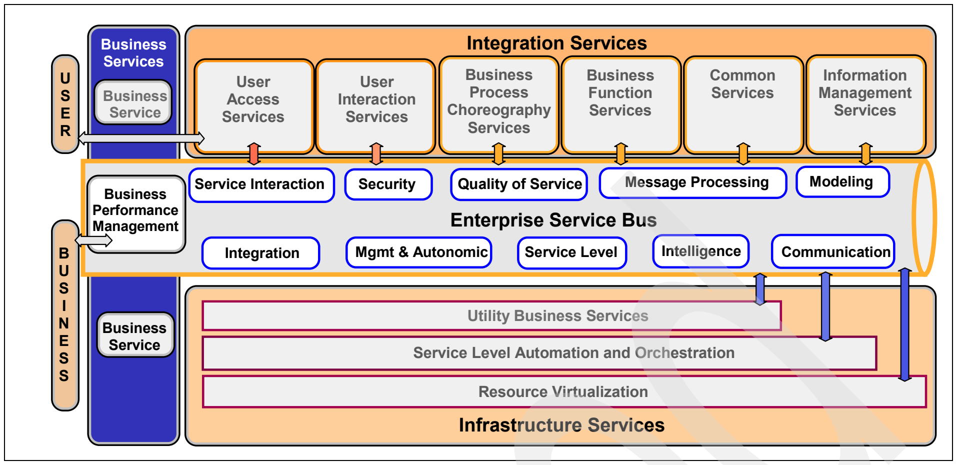


Figure 2-1 Interaction of the ESB with the IT System and main responsibilities

2.1.2 The scope of the ESB

In many traditional IT environments, communication between service requestors and providers is accomplished via a series of direct *point to point* connections, one connection per requestor/provider pair. This requires that the requestor be aware of the requirements of the provider and vice versa, for example, the transport protocol, message format, and location of the service. As enterprise solutions grow, the web of direct connections and the complexity of managing those connections also grows.

In an SOA solution, an ESB acts as an intermediary between requestors and providers. Connections are made to the ESB, rather than directly between the communicating pair. The ESB makes the transition between transport protocols and message formats used by the requestor and provider. The location of each service is known to the ESB but not to the service.

Figure 2-2 on page 8 shows the IBM SOA Reference Architecture and the scope of the ESB.

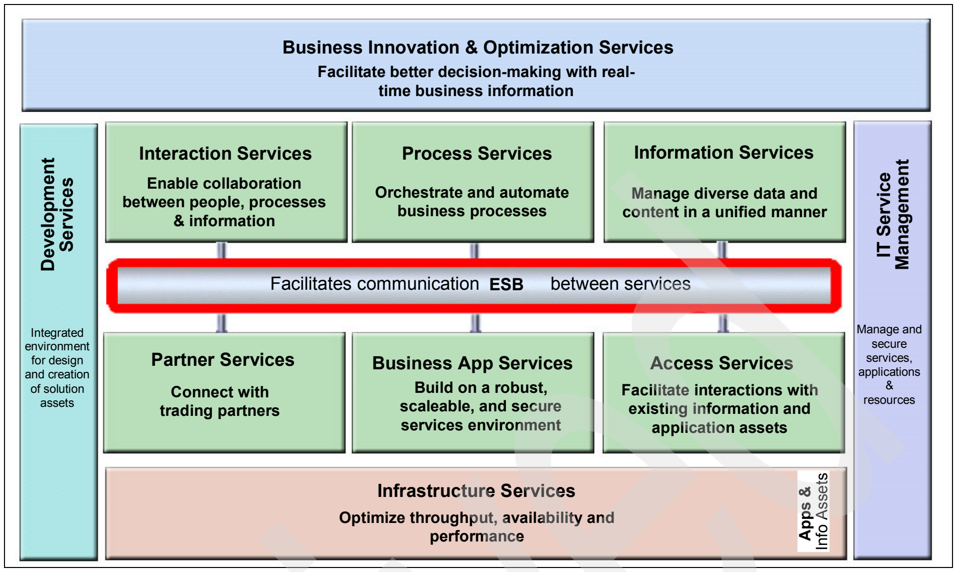

Business Innovation & Optimization Services
Facilitate better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enable collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manage diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
Facilitates communication ESB between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitate interactions with existing information and application assets
Infrastructure Services
Optimize throughput, availability and performance
Apps & Info Assets

Figure 2-2 Scope of the ESB in the IBM SOA Reference Architecture

There is a general misunderstanding that an ESB is primarily a Web Services facility. Definitely, the ESB must be able to accommodate the use of Web Services as part of its functionality. However, the ESB must also be able to support the integration requirements of any enterprise application not designed to use Web Services. At a minimum the ESB must provide transport, mediation, routing, and data handling functionality in order to eliminate the need for coupling between requestors and providers.

Another common misunderstanding is that an ESB can be used to run business processes, including long-running business processes. The ESB should and must be used for the integration requirements of a business process, but the business process model and state should be managed and monitored by a *Business Process Engine (BPE)*. In fact, the ESB provides a connectivity layer for process engines that choreograph the flow of activities between services. It is the ESB responsibility to deliver service requests, routing, rerouting, or transforming them if appropriate.

2.1.3 Basic capabilities of an ESB

In this section we describe the capabilities that an ESB must have in order to fulfil all requirements described in the IBM SOA Reference Architecture and assigned to the bus component. We are not going to focus on a System z solution yet, but give a generic view of an ESB that will help us to identify all responsibilities in its scope regardless of the physical platform.

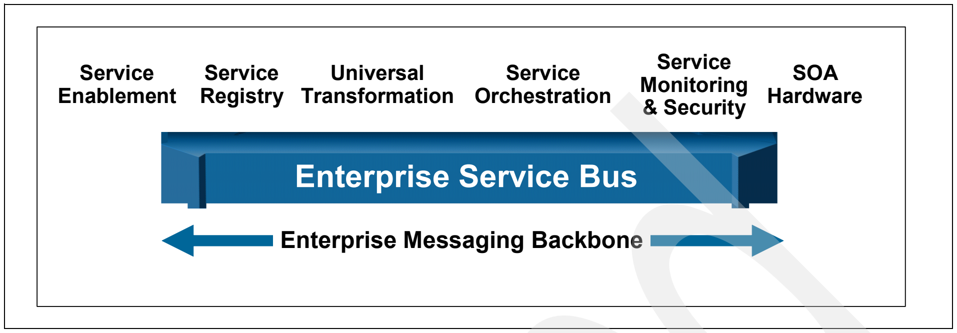


Figure 2-3 Main functionality of an ESB

From the ESB capabilities list we will be able to select different technologies, depending on our requirements, in order to generate the list of requirements that our ESB solution will need.

Table 2-1 lists the capabilities that the ESB must have in order to cover the bus component requirements and provide the minimum functionality needed to support the other components in the SOA.

Table 2-1 Basic capabilities of an ESB

Category and purpose	Capabilities
Communication: Provides location transparency and supports service substitution	► Routing ► Addressing ► At least one messaging style (request/response, publish/subscribe) ► At least one transport protocol that is or can be made widely available
Integration: Supports integration of heterogeneous environments and supports service substitution	► Several integration styles or adapters ► Protocol transformation
Service interaction: Supports SOA principles, separating application code from specific service protocols and implementations	► Service interface definition ► Service messaging model ► Substitution of service implementation
Management: A point of control over service addressing and naming	► Administration capability

In the following sections we explain each capability in more detail.

Communication

The ESB is a communication layer that supplies service interaction functionality. It must support communication through a variety of protocols. It must provide underlying support for message- and event-oriented middleware and integrate with existing infrastructure and other enterprise application integration technologies, such as CICS, IMS, or WebSphere MQ. As a minimum capability the ESB must support at least the protocols that are specifically deployed in the IT environment of the company, as well as those that are generic to the IT industry.

The ESB must be able to route between all these communication technologies through a consistent naming and administration model.

Integration

The ESB supports linking to a variety of systems that do not directly support service-style interactions, or those that were originally not thought to be exposed as services in a heterogeneous environment. This linking includes existing systems, packaged applications, and other integration technologies. These integration technologies might be protocols (for example, JDBC™, FTP, EDI, SOAP, ECI for CICS, OTMA for IMS, and so on) or adapters such as the J2EE™ Connector Architecture resource adapters, DB connectors, CTG for CICS, and so forth. It also includes service client invocation through client APIs for various languages (Java, C++, Cobol) and platforms (J2EE, .Net, CICS/IMS).

Service interaction

The ESB must support SOA concepts for the use of interfaces, and support declaration service operations and quality of service requirements. The ESB must also support service messaging models consistent with those interfaces, and be capable of transmitting the required interaction context, such as security, transaction, or message correlation information.

Management

As with any other infrastructure component, the ESB must have administration capabilities to allow it to be managed and monitored, and to provide a point of control over service addressing and naming. In addition, it must be capable of integration into systems management software.

2.1.4 Extended capabilities for an ESB

The capabilities that were described in the previous section can help assess the suitability of individual technologies or products for implementing an ESB. However, these capabilities only establish which technologies are candidates.

The detailed requirements of any particular scenario drive additional ESB capabilities that can then be used to select specific, appropriate products.

In particular, the following types of requirements are likely to lead to the use of more sophisticated technologies, either now or over time:

- Non-functional requirements related to Quality of Service (QoS) and certain service-level capabilities
- Higher-level Service-Oriented Architecture concepts, such as the use of a service directory, and transformations
- Advanced management capabilities such as sophisticated systems management and autonomic capabilities
- Truly heterogeneous operation across multiple networks, multiple protocols, and multiple domains of disparate ownership

Table 2-2 lists the extended capabilities of an ESB that will help you to identify technologies and products that will be needed in your solution. (This material is presented in a table to save space; the arrangement is not intended to imply any relationship between columns.)

Table 2-2 Extended capabilities of an ESB

Communication	**Service interaction**
► Routing ► Addressing ► Protocols and standards (HTTP, HTTPS) ► Publish/subscribe ► Response/request ► Fire and forget, events ► Synchronous and asynchronous messaging	► Service interface definition (WSDL) ► Substitution of service implementation ► Service messaging models required for communication and integration (SOAP, XML, or proprietary Enterprise Application Integration models) ► Service directory and discovery
Integration	**Reliability**
► Database ► Existing and application adapters ► Connectivity to enterprise application integration middleware ► Service mapping ► Protocol transformation ► Data enrichment ► Application server environments J2EE and .Net) ► Language interfaces for service invocation (Java, C/C++/C#)	► Transactions (atomic transactions, compensation, WS-Transaction) ► Various assured delivery paradigms (WS-ReliableMessaging or support for Enterprise Application Integration middleware)

Security	**Service level**
► Authentication ► Authorization ► Non-repudiation ► Confidentiality ► Industry security standards (Kerberos, WS-Security)	► Performance (response time, throughput, and capacity) ► Availability ► Other continuous measures that might form the basis of contracts or agreements
Messaging processing	**Management and autonomic**
► Mediation ► Encoded logic ► Content-based logic ► Message and data transformations ► Message/service aggregation and correlation ► Validation ► Intermediaries ► Object identity mapping ► Service/message aggregation ► Store and forward	► Administration capability ► Service provisioning and registration ► Logging ► Metering ► Monitoring ► Integration to systems management and administration tooling ► Self-monitoring and self-management
Modeling	**Infrastructure intelligence**
► Object modeling ► Common business object models ► Data format libraries ► Public versus private models for business-to-business integration ► Development and deployment tooling	► Business rules ► Policy-driven behavior, particularly for service level, security and quality of service capabilities (WS-Policy) ► Pattern recognition

In the following sections we explain each capability in more detail.

Communication

The ESB solution selected might be required to support certain communication protocols and styles, such as synchronous and asynchronous, trigger definitions, and so forth. It is important to make sure that all communication requirements used by solutions currently running are looked at. That way you will avoid handling transformations in the Business Application services layer.

Service interaction

It is recommended that the ESB has or has access to a repository structure where all service declarations are kept. This way, for each service request the repository can be accessed to identify which service instance best fits the request and the information needed to execute the call can be generated. This repository could manage the status of the services, different releases, syntaxes and communication protocols supported by each service.

Integration

The ESB should be capable of connectivity to a wide range of different service providers, using adapters and integration middleware. It should be capable of data enrichment to alter the service request content and destination on route, and map an incoming service request to a one or more service providers.

Quality of Service (QoS)

The ESB might be required to support service interactions that need different Qualities of Service to protect the integrity of data mediated through those interactions. This support can involve transactional support, compensation, and levels of delivery assurance. These features could be variable and driven by service interface definitions. Other Quality of Service considerations for an ESB include:

- Support qualities of service on top of communication protocols that are fundamentally more brittle
- Business transactions that span several systems that need to be monitored as a whole
- Support for exception and error handling

Security

The ESB could ensure the integrity and confidentiality of the services that it carries is maintained. It could integrate with the existing security infrastructures to address the essential security functions, such as:

- Identification and authentication
- Access control
- Confidentiality
- Data integrity
- Security management and administration
- Disaster recovery and contingency planning
- Incident reporting

Additionally, the ESB integrates with the overall management and monitoring of the security infrastructure. The ESB can either provide security directly or can integrate with other security components, such as authentication, authorization, and directory components.

Service level

The ESB mediates interactions between systems supporting specific performance, availability, and other requirements. It offers a variety of techniques and capabilities to meet these requirements. The ESB provides support that allows technical and business service level agreements to be monitored and enforced. It integrates with the corporate system tools to allow you to create a comprehensive environment to measure level service agreements.

Message processing

The ESB must be capable of integrating messages, objects, and data models between the application components of an SOA architecture. It could also be able to make decisions, such as routing, based on the content of service messages. The ESB needs a mediation model that allows message processing to be customized. The model also allows sequencing of infrastructure services (for example, security logging and monitoring) around business services invocations. Mediations can be located close to consumers, providers, or anywhere in the ESB infrastructure that is transparent to consumers and providers. Mediations can also be chained. The ESB validates content and format.

The ESB provides digital certificates control of validation, connecting to certification authorities. It could have semantic and syntax validations of messages. This functionality will avoid sending incorrect messages to business components and creating overwork in these units.

Management and autonomic

In addition to basic management capabilities, the ESB also supports the migration to autonomic and on-demand infrastructure by supporting metering and billing, self-healing and dynamic routing, and it could be able to react to events to self-configure, heal, and optimize. It integrates with the infrastructure layer to take advantage of the utilities offered, like dynamic capacity of process or memory, high performance routing capabilities, and so forth.

Modeling

The ESB supports the increasing array of cross-industry and vertical standards. It supports custom message and data models, especially the ones that are currently used, so it will avoid having to make modifications on the Business Application services. The ESB also supports the use of development tooling and should be capable of identifying different models for internal and external services and processes.

Infrastructure intelligence

The ESB will be capable of evolving towards a more autonomic, on-demand infrastructure. It allows business rules and policies to affect ESB function, and it could support pattern recognition.

2.2 Key architectural decisions with respect to the ESB

From the moment that an ESB implementation is considered, immediately a number of architectural questions will come up. In this section we mention most of the common architectural decisions that must be made with regard to the ESB; this list can serve as an outline for the rest of this chapter. The following questions must be answered:

- What business solutions and services need to be connected to the ESB, either as a requestor or a provider?
- What specific industry communication protocols need to be supported by the ESB?
- What platforms will the ESB need to run on and how many physical instances will be needed?
- What kinds of data transformations need to be supported by the ESB?
- What types of connectors, if any, need to be supported by the ESB?
- Are synchronous, asynchronous or both styles are needed?
- What type of mediation is required by the business services?
- Are publish/subscribe rules required?
- How will the following non-functional requirements be supported?
  - Security (authentication, authorization, encryption, and so on)
  - Level of availability
  - Response times expected
  - Throughput (number of messages, transformations, mediations per time unit)
  - Reliability (transactionality, 2-Phase Commit, compensation)
  - Maximum number of concurrent requests
  - Development tooling capabilities
  - Monitoring and systems management infrastructure
  - Scalability of the ESB
- Who will manage the ESB?

2.3 Common patterns

In this book our primary focus is to provide guidance on the platform decision and to clarify the differences between implementing an ESB on System z or elsewhere. A good approach for considering the platform decision is to review a number of common patterns. There is a good chance that you very quickly will conclude that one certain pattern is the right one for your situation.

In this section we describe the most common architectural patterns. We focus mainly on how to integrate the different business solutions with the ESB and on which platform the ESB can best offer its functionality to the other services in the SOA.

We have identified two kinds of patterns that are helpful in determining the platform:

- **Location patterns:** The first set of patterns directly relates to the platform location of the ESB and its integration with business solutions deployed on a variety of other platforms in the enterprise.
- **Communication patterns:** The second set of patterns is how the business applications connect to the ESB, what kind of connectors they need to use, and what kind of communication protocols and language they require. These patterns help more in determining which type of ESB is required.

Other patterns for the ESB (not commented on further here) are:

- **Interaction patterns:** This refers to how the request/response style interactions between the ESB and business solutions are handled. One variation is when more than one response or an “ack” message can be sent to a service requester.
- **Mediation patterns:** These patterns consider how service requestors are enabled to dispatch their messages using a variety of interaction protocols or APIs (for example, SOAP/HTTP, JMS, and MQI) by transcoding requests into a form that is more digestible by the targeted service provider.
- **Aggregated patterns:** Mediation and interaction patterns can be combined to realize more complex patterns.

Since we consider these last three pattern types to have no influence on the selection of the platform, we only focus on the location and communication patterns. More information regarding ESB patterns can be found in the Redbooks publication *Patterns: Implementing an SOA Using an Enterprise Service Bus*, SG24-6346.

Important: When we refer to “System z” we are talking about the hardware platform, regardless of the operating system. This means that the comment applies to both z/OS and Linux on System z.

When we refer to a “distributed platform” we are talking about x86 and UNIX/RISC architecture based server platforms. We do not specify any specific operating system.

2.3.1 Location patterns

For location patterns we are interested in the location of these three elements:

- The service requestor
- The ESB itself
- The service provider

We identified two possible locations for each element:

- The System z platform
- Any type of distributed platform, such as x86 or a RISC/UNIX® architecture

We define the various location patterns based on where each of the three elements is located.

The general idea of the location patterns is depicted in Figure 2-4.

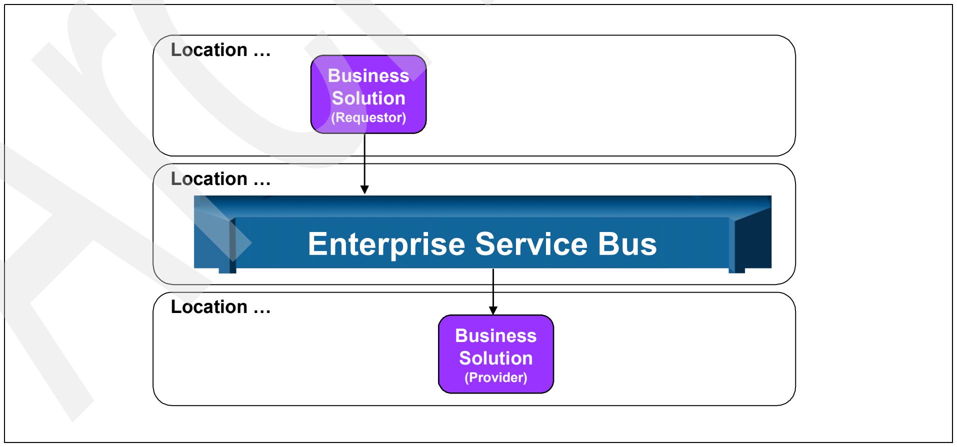


Figure 2-4 Location patterns for the ESB

In the following sections we discuss these patterns:

- Requestor on distributed - ESB on distributed - Provider on System z
- Requestor on System z - ESB on distributed - Provider on System z
- Requestor on System z - ESB on System z - Provider on System z
- Requestor on distributed - ESB on System z - Provider on System z
- Requestor on distributed - ESB on System z - Provider on distributed
- Requestor on distributed - Federated ESB - Provider on System z

Important: We recommend that you read through the various patterns sequentially because in each pattern we talk about specific aspects of an ESB implementation on System z versus distributed.

Requestor on distributed - ESB on distributed - Provider on System z

This is probably the most commonly deployed location pattern for an ESB. This pattern is typically used if applications on distributed platforms need to retrieve information from applications running on System z, such as CICS, IMS, and DB2. For instance, you could have a J2EE solution deployed on a UNIX system that connects to an ESB which is also installed on UNIX and using connectors to access, for example, CICS.

Figure 2-5 depicts this pattern.

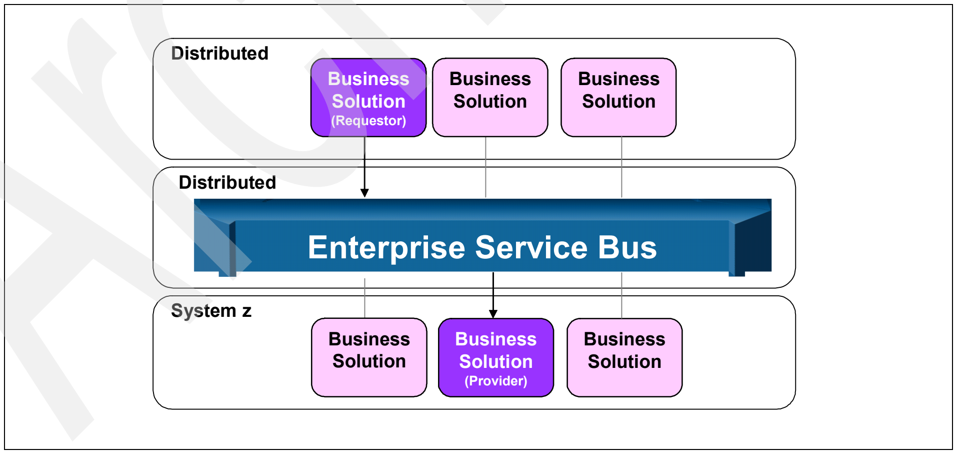


Figure 2-5 Requestor on distributed - ESB on distributed - Provider on System z

When to use

This pattern can be used if your back-end services run on System z and your service requestors (J2EE, .Net) run mostly on distributed machines, and you consider the Quality of Service offered by the distributed platform adequate for the ESB. A condition for this pattern is that the distributed ESB provides adequate connectivity with the back-end services on System z.

When not to use

There are a number of reasons to not use this pattern, mostly related to QoS issues. This pattern is not appropriate if you have:

- Very strict reliability requirements, such as 2-Phase Commit on certain mediations.

  The z/OS operating system on System z offers the best options for reliable mediations and transformations. If, for example, many database updates are triggered from a single mediation, a bullet-proof 2-Phase Commit mechanism becomes essential. This is even more important in certain industry solutions, such as banking.

- Extremely high availability and scalability requirements.

  Like normal business application services, an ESB might also have extreme availability and scalability requirements. Because the ESB is the backbone of all communication between services, it is not a surprise that the ESB ends up having the most stringent availability and scalability requirements. A “5 nines” availability and capability to deal with extreme peaks is not uncommon for an ESB. System z (z/OS more so than Linux on System z) offers superior scalability within one single box or multiple boxes dispersed over multiple locations, if desired.

- Very strict requirements with respect to security.

  ESB implementations on distributed platforms very easily result in the use of multiple servers connected through the network. Those additional connections introduce additional points of security risk. System z and especially the z/OS operating system has a very sophisticated security model, making it very difficult to tamper with. Also, it is easier to run a complex and high workload ESB within one System z box by making use of virtualization techniques to simulate multiple logical servers (LPARs).

- An expectation that the majority of your service requestors will be on System z.

  If, in addition to your service providers and data, the majority of your service requestors are expected to run on System z as well, it might be more efficient to consider the pattern discussed in “Requestor on System z - ESB on System z - Provider on System z” on page 22.

- Response times that are critical.

  Proximity of services and data to the ESB is important for achieving acceptable response times and keeping the path length short. Cross-memory access from the ESB to a Database or Transaction server is obviously faster and more efficient than access over a network.

> **Important:** Note that access from the service requestor to the ESB is usually only one asynchronous call (either over HTTP or JMS). Access from the ESB to the service provider is usually a series of multiple sequential or parallel calls, either synchronous or asynchronous, and eventually with a transactional context. So, in a discussion about path length and response times, access from the ESB to the service providers is much more important than access from the service requestors to the ESB!

Requestor on System z - ESB on distributed - Provider on System z

Of course, existing System z environments already have numerous application components communicating with each other. Traditional communication on System z is done using cross-memory techniques, TCP/IP, or still in some cases through proprietary protocols, such as APPC.

This pattern typically exists in environments where there is a System z landscape and where there is already an ESB running on a distributed platform for use by distributed applications. This pattern is depicted in Figure 2-6.

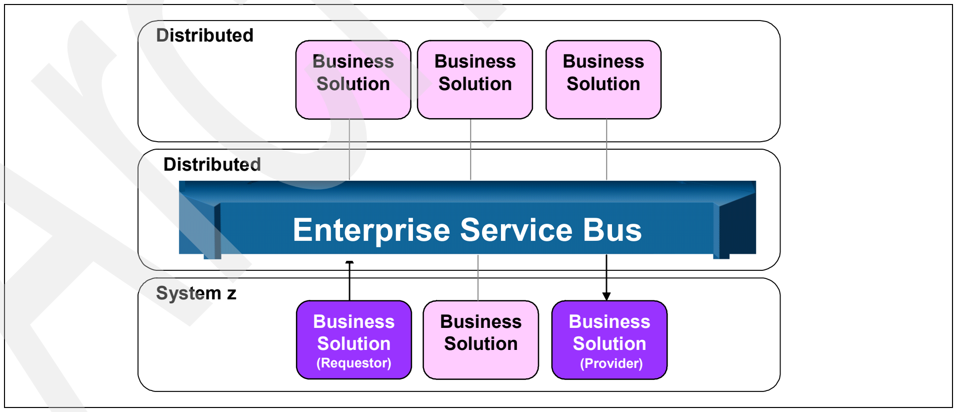


Figure 2-6 Requestor on System z - ESB on distributed - Provider on System z

When existing System z applications need to integrate through an ESB replacing the traditional point-to-point integration there is an option to use the ESB already implemented on distributed. Technically and functionally, having a service requestor and service provider on System z and “stepping out” to a distributed platform to use ESB functionality should not be a problem, as long as the required connectors, communication protocols, and data formats are supported. However, issues with this pattern might occur that relate, once again, to achieving an adequate level of Quality of Service and with fulfilling the NFRs. The optimal way to integrate between service requestors and service providers on System z would be to use an ESB on System z as well. This would reduce the path length, minimize security exposures, and enhance RAS.

One way to deal with the situation of having many service requestors and service providers on System z and an ESB already implemented on distributed could be to implement a so-called “federated” ESB on System z. This federated ESB on System z can then be logically integrated with the existing ESB on distributed. This architecture would work as follows:

- Distributed service requestors continue to connect to the ESB on distributed.
- System z service requestors connect to the ESB on System z.
- Service providers on distributed accessed by service requestors on distributed use the ESB on distributed.
- Service providers on System z accessed by service requestors on System z use the ESB on System z.
- Service providers on System z accessed by service requestors on distributed can be accessed by the ESB on distributed directly or the request can be routed via the ESB on System z.
- Service providers on distributed accessed by service requestors on System z can be accessed by the ESB on System z directly or the request can be routed via the ESB on distributed.

The architecture of a multi-platform federated ESB will work out best if the ESB software chosen is the same on both distributed and System z. That way, even though there are multiple runtime platforms, development and deployment tasks will be the same.

When to use

This pattern is valid in those situations where there are both service requestors and service providers on System z, but there is no intent to implement an ESB on System z and the Quality of Service (scalability, availability, security, response times) offered by the ESB on the distributed platform is considered adequate. Furthermore, the ESB on distributed must support all protocols and connectors being used by the service requestors and service providers on System z.

When not to use

Do not choose this pattern if System z is considered a strategic platform and the additional QoS of the System z platform is required by the ESB.

Requestor on System z - ESB on System z - Provider on System z

This is a pattern that might be already implemented on System z, even if there is no dedicated ESB product implemented. In certain cases, ESB-like functionality is implemented in custom-built frameworks, WebSphere MQ exits, or within business logic. In some cases an ESB product, such as WebSphere Message Broker, is already deployed on System z. This pattern is shown in Figure 2-7.

In this book we do not elaborate any further on the value of obtaining and implementing a dedicated ESB solution, but the recommended approach is to investigate whether a specialized ESB product should replace the existing integration technology. Later in this chapter we discuss ESB solutions and products.

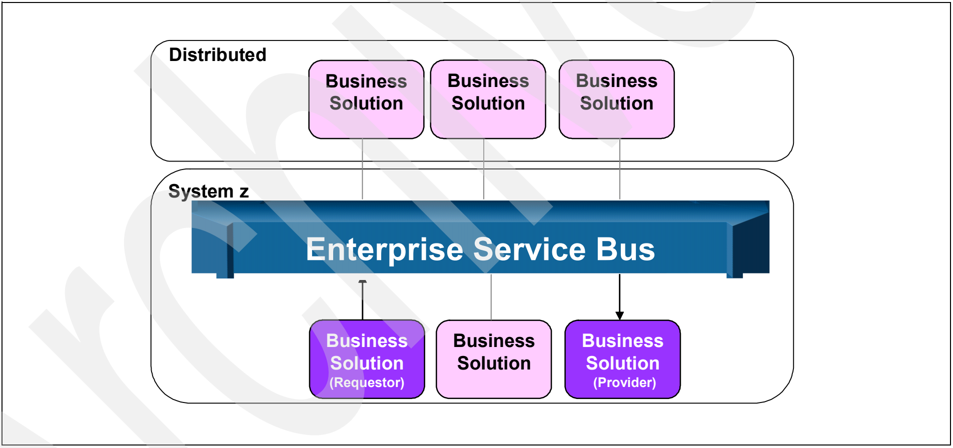


Figure 2-7 Requestor on System z - ESB on System z - Provider on System z

When to use

This pattern is especially recommended if the majority of your service requestors and service providers reside on System z and you wish to benefit from the enhanced QoS of an ESB implementation on System z.

When not to use

We can assume that if there is an increasing trend of service requestors and service providers on System z, System z is a strategic part of the SOA. Apart from political and organizational constraints, there are not many reasons to *not* implement the ESB on System z. If a lot of time and money already has been invested in implementing an ESB on distributed, the QoS benefits of having an ESB on System z as well might not outweigh the additional investment of placing an additional ESB on the System z.

Note: IBM WebSphere ESBs on System z and distributed are functionally equivalent and very similar from an administration point of view. From a skills perspective it is not a big step to expand a WebSphere ESB implementation on distributed to System z.

IBM WebSphere Message Broker on System z and distributed have differences from a functionality point of view, but administration is very similar on both System z and distributed.

For both ESB solutions, obviously, the systems programming is significantly different.

More information about those ESB solutions follows later in this chapter.

Requestor on distributed - ESB on System z - Provider on System z

This interaction is very similar to the pattern discussed in “Requestor on distributed - ESB on distributed - Provider on System z” on page 18; however, in this pattern the ESB is implemented on System z. The service requestors (on distributed) connect to the ESB on System z over the network and the ESB will access the service providers on System z locally, either cross-memory, using HiperSockets™, or via TCP/IP. This pattern is illustrated in Figure 2-8 on page 24.

For the service requestors on distributed it is actually fully transparent where the ESB resides because the protocols typically used to connect to the ESB are HTTP, JMS, SOAP over HTTP or SOAP over JMS. With many service requestors scattered over many distributed machines it actually does make sense to have the ESB implemented in a more centralized environment. We must not forget that in a real SOA the ESB will become the layer with the highest scalability and availability requirements, and over time, also the layer with the highest security requirements.

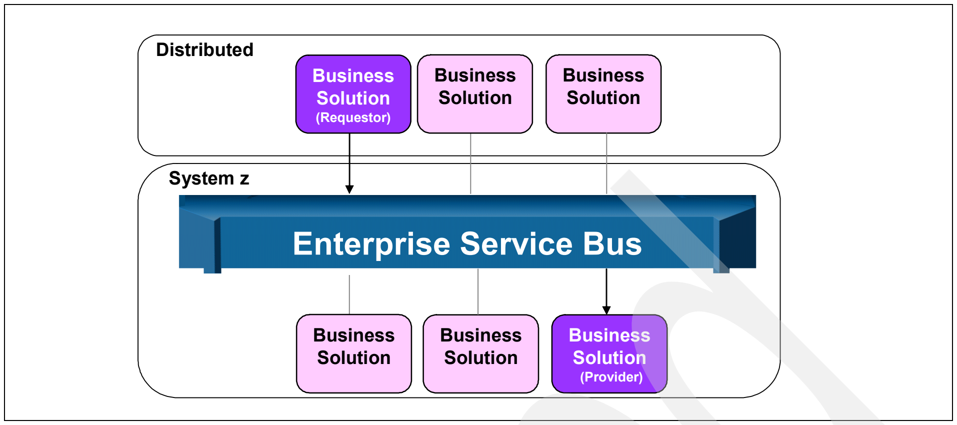


Figure 2-8 Requestor distributed - ESB System z - Provider on System z

When to use

This pattern really makes sense if most of the service providers are on System z and growing, and there is a strong need for the robustness of an ESB on System z. Another reason could be if very specific connectors between the ESB and certain service providers are required. Examples of this are the VSAM nodes and the CICS EXCI nodes in WebSphere Message Broker on System z (z/OS only). These nodes do not exist in WebSphere Message Broker on the distributed platform.

Also, an ESB on System z will have a more centralized character from physical infrastructure, systems management, and operability perspectives. This is generally a plus for a complex and large scale ESB implementation.

When not to use

Again, the same arguments discussed for the Requestor on System z - ESB on System z - Provider on System z pattern would be true. Additionally, if certain service requestors on distributed have very specific needs that require a local or specific connector into the ESB that is not available on System z, an ESB on System z might become less compelling, even if most of the service providers are on System z. Note that some incoming data formats and protocols might only be available in the chosen ESB product on distributed, but not on System z.

Requestor on distributed - ESB on System z - Provider on distributed

So far we have only talked about patterns in which the service providers reside on System z. In this pattern the service providers as well as the service requestors are on distributed. In this case, does it make sense to have the ESB on System z?

Even if the service requestors, service providers, and the ESB are all on distributed, it still takes network calls for both the requestors and providers to communicate with the ESB, which will most likely reside on another server than any of the service requestors and providers. Furthermore, because of the way high availability works on most distributed machines, a logical interaction between a requestor or provider and the ESB might even spark multiple “hops” over the network, reducing efficiency, response times, and increasing security risks.

Important: Note that a large scale enterprise ESB implementation on distributed *will* result in the implementation and management of several to many physical machines, depending on the operating system (Windows®, Linux on X86, UNIX, AIX®, and so on) and the hardware platform. Routers, switches and intelligent workload balancers will be necessary to route a service request to a service provider in an efficient way. Also, more and more systems management software and personnel will be needed to keep an eye on all of this because the ESB will keep on growing.

In this pattern, with an ESB on System z, you will always have only two network hops, one from the requestor on distributed to the ESB on System z and one from the ESB on System z to the provider on distributed. All load balancing and routing between instances of the ESB on System z is dealt with internally on System z, using high performing communication techniques, such as cross-memory (if ESB and requestor/provider are on the same LPAR), XCF (if ESB and requestor or provider are on different LPARs or even on different machines) or HiperSockets (for instance, if a requestor/provider is on Linux on System z and the ESB is on z/OS).

Important: An ESB is like storage: it will keep on growing indefinitely once it is there and once the value is experienced. Choosing a platform with supreme scalability characteristics and flexibility to grow significantly is essential for an ESB.

This pattern is depicted in Figure 2-9 on page 26.

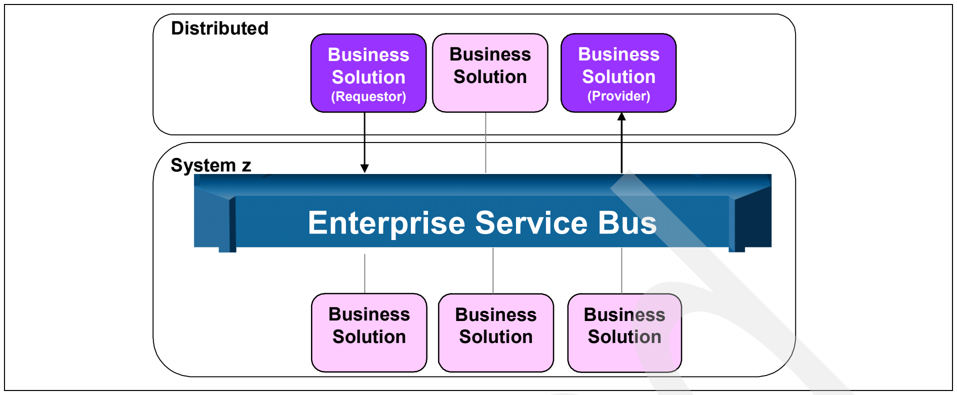


Figure 2-9 Requestor on distributed - ESB on System z - Provider on distributed

When to use

This pattern makes sense if the QoS offered by System z is essential for the ESB implementation. From a functionality perspective there is not really a reason to locate the ESB on System z. Some ESB solutions do offer unique functions on System z, such as the previously mentioned VSAM and EXCI nodes in WebSphere Message Broker on System z (z/OS only), but those functions will not be required if all service requestors and providers reside on distributed.

When not to use

One of the most common reasons to not use this pattern is if the service requestors and providers on distributed require specific connectors, communication protocols, and data formats that are not supported in the specific ESB product or solution chosen on System z. Another reason could be if the System z platform has been declared non-strategic, as would be the case if significant investment has already been made in an ESB on distributed or an ESB product has been chosen that is not available on System z.

Requestor on distributed - Federated ESB - Provider on System z

In this context, by “federated” we mean an ESB that is partly implemented on distributed and partly on System z. Depending on the product chosen, an ESB domain can be established with different points of entry and exit on both distributed and System z. WebSphere MQ is an excellent solution to handle cross-platform routing of service requests.

We feel that this type of ESB implementation is the best approach for companies with many potential service requestors and service providers on both distributed and System z. This way the advantages of an ESB on distributed, which are typically function, and the advantages of an ESB on System z, which typically involve Quality of Service, can be molded together and a service requestor/provider interaction can be routed using the most optimal route, depending on function and QoS requirements.

This pattern is shown in Figure 2-10, in this case with the service requestors typically on distributed and the service providers typically on System z.

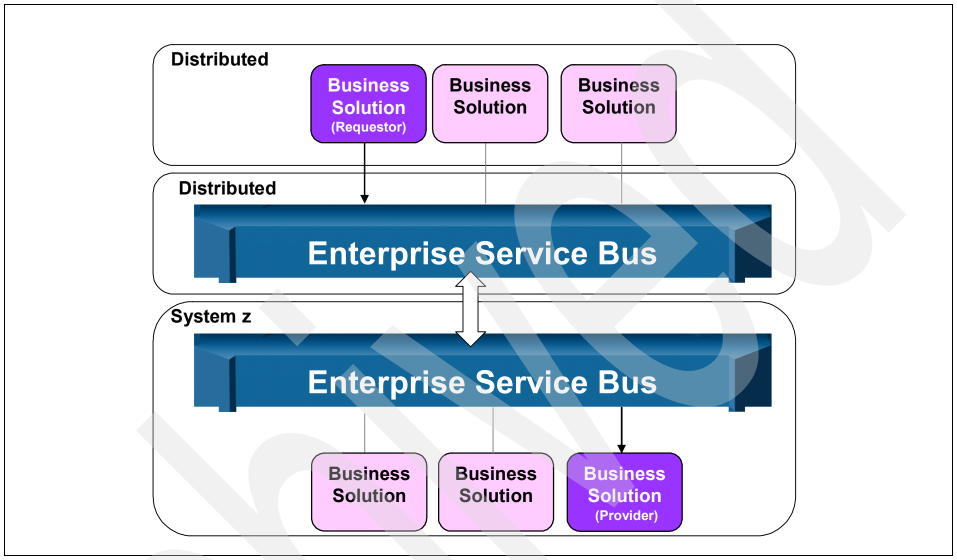


Figure 2-10 Requester on distributed - Federated ESB - Provider on System z

How Federated ESB works

In an SOA it is very common that a request sent by a service consumer requires resources from multiple service providers on multiple platforms and connected to multiple ESBs (also on multiple platforms). One option to deal with such requests is to let the ESB to which the service consumer initially connects coordinate the entire process, including all mediations. This option is adequate as long there are only a few occurrences of this type of request. If there are numerous occurrences of this type of request, this option would lead to federation logic scattered across multiple ESBs. Also, redundancy is very likely to occur in similar mediations implemented in multiple ESB hubs. This way we will end up having higher costs to maintain and extend the ESB implementation. To avoid this complexity it is recommended that each ESB hub mediate its own service requests that operate with service providers directly connected to this ESB hub, and additionally, that a main ESB hub component be used that will mediate access to service providers not directly connected to the initial ESB hub.

The key characteristic is to encapsulate the heterogeneous ESBs' implementation complexity inside the ESB global solution, thereby making it transparent to business services involved, consumers, and providers.

It is the responsibility of the architecture team to integrate multiple ESBs and to configure them to interchange messages, giving the impression to applications and application architects that they are connecting to a single ESB system.

When to use

This solution is highly recommended in cases where there are many service requestors and service providers on both distributed and System z, as well as a reason to have ESBs on both distributed (probably for functionality reasons) and System z (for functionality and QoS reasons).

When not to use

A federated ESB does not make sense if service requestors and service providers only exist on one platform, either distributed or System z. In that case, a federated ESB across both distributed and System z might be too complex.

2.3.2 Communication patterns

Another category of patterns is "communication" patterns. These patterns influence directly which ESB product or solution is the best one, but also indirectly influence the platform that the ESB should run on. Some communication protocols, connectors, and adapters might not be available on all platforms or they might provide different Quality of Service levels on different platforms.

When we talk about *communications* we are referring to the way applications connect to the ESB and how the ESB operates with these applications. This includes connectivity, semantics, and syntax of messages.

An ESB offers different ways to connect to the ESB, based on *connectors* or *adapters.* There are different types of connectors. Some of them are generic and follow standards, allowing any application that follows those standards to use them, and some of them are specific for certain kinds of ISV application solutions, such as SAP®, PeopleSoft® and so on.

This principle is shown in Figure 2-11 on page 29. Note that this diagram does not refer to any ESB product specifically, but rather shows a number of common ISV solutions and communication mechanisms you might expect in an ESB solution.

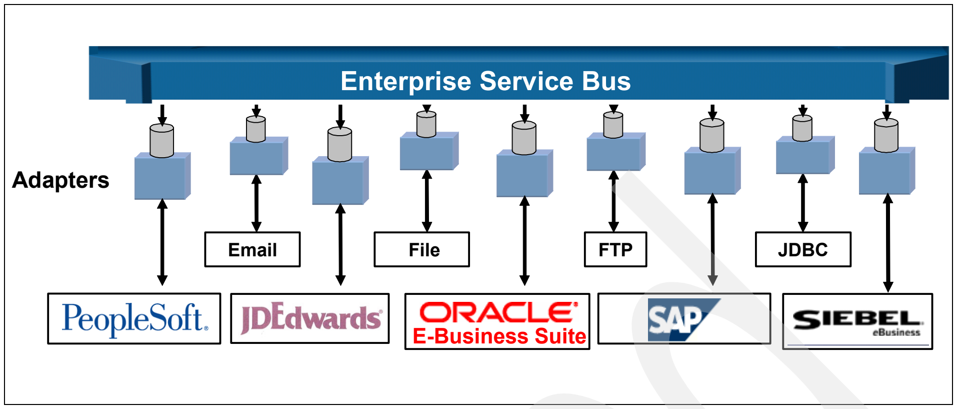


Figure 2-11 ESB connector architecture

Many times, business applications have more than one option to connect to the ESB because they support both specific connectors using a specific protocol or API as well as a generic connectors compliant with industry standards. Depending on factors like high availability and performance requirements, skills of the support team, and so on, it is recommended to choose one type of connector or another.

Specific connector communication pattern

Examples of business solution specific connectors are the SAP connector/adapter for SAP access, CICS Transaction Gateway for CICS, and IMS Connector for Java for IMS. These connectors have matured significantly over time, are well tuned, and offer the best Quality of Service (performance, security, scalability, and availability). They also permit application architects to focus on building business services using their own data format and communication protocols rather than having to conform to generic standards.

A good ESB solution will abstract the specific connectors and adapters from the rest of the SOA landscape and perform transformations, if necessary. A service consumer can easily use a generic ESB connector, for example, a SOAP node in WebSphere Message Broker, to connect to the ESB and finally access SAP through an SAP-specific adapter.

Important: For a successful SOA there is no need to transform all communication into Web Services standards. The success of an SOA is determined by how well an ESB can abstract and transform protocols and data formats. Both service requestors and providers must remain able to use their own communication protocol and style and data format, independent of the other party.

For example, a service requestor can send a request to the ESB using the SOAP over HTTP protocol, in XML data format and in asynchronous style. The service provider might be a CICS program only supporting proprietary CICS access. In that case the CICS program can be accessed from the ESB using the EXCI communication protocol, a COMMAREA data format and in synchronous style. In this example the ESB would perform both protocol conversion and data format conversion, as well as manage the fact that the handshake between the service requestor and the ESB is asynchronous and the handshake between the ESB and the service provider is synchronous.

When to use

This pattern should be used when a certain QoS is required that is not sufficiently available from generic connectors. Examples are high performance, 2-Phase Commit, and the use of certificates to authenticate to the service provider. Generic connectors will always have certain overhead compared to specific connectors and therefore offer lower performance.

Another reason to use this pattern is when support for generic connectors is simply not available in the service requestor or service provider runtime container. There are still many ISV solutions that only support a specific connector.

When not to use

Using generic connectors simplifies the infrastructure and lessens the skills requirements. When the strategic approach is to have homogenous communication and there is no requirement for high performance or specific QoS, the use of specific connectors should be avoided. Increasing the use of generic connectors (typically Web Services and XML based) generally reduces the amount of protocol and data format transformations in the ESB. However, it will increase the overhead on the service provider side.

Generic connector communication pattern

In SOA, generic connector communication typically means using Web Services. Web Services can be used between the service requestor and the ESB and between the ESB and the service provider. It is also possible to use Web Services for communication, for example, between a Business Process Engine

(BPE) and an ESB, between a WebSphere Portal and an ESB, or even between an ESB and another ESB. When using Web Services, the service requestor or service provider or both need to support this. One of the first things to check out is which of your business solution runtime containers (for example, CICS, IMS, TSO, IDMS, and so on) support Web Services natively.

The advantage of this pattern is that complexity and overhead in the ESB will be lower because service providers can directly work with the protocol and data format used by the ESB itself and, in most cases, the service requestor. The disadvantage is that business solutions will not be able to communicate with the ESB using its own native data formats and protocols, and the ESB will therefore have to transform or build the internal messages internally upon arrival of the generic message.

A note on QoS and Web Services: Web Services has matured significantly with regard to Quality of Service. WS specs are available and supported in IBM's middleware for 2-Phase Commit and security functions. However, the maturity and quality of those specs is not exactly the same as the "native" QoS support implemented in specific connectors running on z/OS. Furthermore, some additional overhead will occur when using the additional WS specs. In specific connectors, the amount of overhead is minimal when using, for example, 2-Phase Commit.

When to use

Reasons to use this pattern are:

- When there is no specific connector available in the ESB for the runtime container of the service provider or service requestor and the service provider or service requestor supports Web Services.
- When IT Architecture is strongly pushing the usage of homogeneous, standardized communication – for obvious reasons, such as IT infrastructure simplification and skills availability.

When not to use

Reasons for not using this pattern could be:

- When you have specific QoS requirements, such as high performance, 2-Phase Commit, or sophisticated security.
- When no support is available for Web Services in the service requestor or service provider.

2.3.3 Reasons not to use an ESB

It is not a firm requirement to use an ESB at all times. Although using an ESB is the default choice, an exception path must be available for certain situations. So, what are those exceptional situations? In this section we discuss scenarios where avoiding an ESB can produce better results. These scenarios involve the following issues:

- Business application services run in the same container
- Components are tightly coupled by nature
- End-to-end 2-Phase Commit
- High performance
- Security
- Cost

Important: Whenever the ESB is bypassed and another communication/integration technology or mechanism is used, this can complicate your IT infrastructure and increase skills requirements to maintain the solution. Strict governance should be exercised to bypass the ESB only when absolutely necessary.

Business application services run in the same container

If you have business application services that run in the same container (for instance, two Java services that are part of the same business application and that run in the same runtime container, such as WebSphere Application Server), there is a performance benefit when you bypass the ESB. The question becomes whether this performance benefit outweighs the loss of standardized communication and flexibility through the ESB. If extreme performance and throughput are required, however, the decision to not use an ESB might be justified.

Components are tightly coupled by nature

Tight coupling can occur when business functions made for different types of users are built as components of one whole business application, where they share internal resources, such as business components, common functions, and so on. They might even share part of their database schema. It might be more effective to integrate these business functions directly rather than using an integration layer that is not going to contribute any valuable functionality. This does not mean that those business functions should not also be published as Business Application services to the ESB, since the services might be used by other business applications that are not part of the same application.

End-to-end 2-Phase Commit

Transactions and *2-Phase Commit* are fundamental concepts in building reliable applications. Running a transaction in a 2-Phase Commit mode provides a mechanism to ensure that all the participants in the transaction achieve a mutually agreed outcome. Traditionally, transactions have the following properties, collectively referred to as *ACID*:

- **Atomicity**
  If successful, then all the operations happen, and if unsuccessful, then none of the operations happen.
- **Consistency**
  The application performs valid state transitions at completion.
- **Isolation**
  The effects of the operations are not shared outside the transaction until it completes successfully.
- **Durability**
  Once a transaction successfully completes, the changes survive failure.

From the moment a 2-Phase Commit transaction starts, data blocks are allocated and locks are placed on data that will be updated. This is done to make sure that at the end of the transaction, just before making the final updates, the state of the data is still the same as at the beginning of the transaction. This situation should exist for the shortest period of time possible, typically only for a few hundreds of a second (common in CICS and IMS transactions on z/OS) or a few tens of a second at the most. This concept conflicts with the thought of “loosely coupled” systems with an ESB in the middle. Careful thought must be given when designing an SOA that includes an ESB and that processes 2-Phase Commit transactions.

ESB solutions might offer support for transactions and 2-Phase Commit; however, this usually means that the mediation is executed as a transaction and rollbacks or compensation transactions are executed in case an unexpected error occurs during the execution of the mediation inside the ESB. IBM’s WebSphere ESB and WebSphere Message Broker offer this function. An ESB cannot, on its own, manage an end-to-end transaction using 2-Phase Commit from the moment the service requestor sends the request to the time the service provider has successfully sent back the response. In an STP process, fully executed under 2-Phase Commit (for example, a payment process), an ESB would not be a good choice because the ESB will “break” the 2-Phase Commit scope. What you will be able to achieve is independent 2-Phase Commit transactions:

- Between the send of the request by the service requestor and the receipt of that request by the ESB, if both the requestor and the connector used in the ESB support 2-Phase Commit. This can be by means of, for example, support for XA interfaces, WS-Atomic Transaction or a connector using Resource Recovery Services (RRS) on z/OS.

- For the mediation in the ESB, in case the ESB product or technology supports 2-Phase Commit for mediations.
- Between the outgoing node in the ESB and the receipt of the message by the service provider, if both the connector in the ESB and service provider support 2-Phase Commit. This can be by means of, for example, support for XA interfaces, WS-Atomic Transaction, or a connector using Resource Recovery Services (RRS) on z/OS.

It is very important to understand the entire scope of a 2-Phase Commit transaction, in other words, where it starts and ends, to identify the impact of an ESB.

High performance

An ESB will usually not provide better performance than a direct point-to-point interface between a requestor and provider that is designed for performance and high throughput. This difference in performance depends a lot on how much abstraction is implemented in the ESB and how much transformation and routing logic is implemented in the mediation. Depending on the exact ESB product, transformation can add significant overhead and more resources will be required to get the same performance as a direct point-to-point interface. In some cases a bottleneck in throughput might be reached in the ESB, which can only be lifted by expanding the ESB with additional hubs, adding servers, or both. By the way, should this situation occur, it is easier to scale up an ESB implementation on z/OS than it is on other platforms. We get into this in more detail later in this chapter.

The bottom line is that interfaces that absolutely cannot compromise performance and that do not absolutely need the intelligence of an ESB might qualify for an exception and be implemented outside the ESB. This is even more so the case if those characteristics are combined with the ones discussed in the previous three subsections. A good example of this is a payment transaction in a banking environment.

Security

Although ESB solutions are offering increasing security features and functions, the specifically chosen ESB solution or product might not be secure enough to publish all services on. Some services might require such specific and sophisticated security that they can only be accessible by a highly secure direct connection using a proprietary protocol or a common protocol, but highly secure (using a soft or hard token to authenticate, encryption, or even PKI).

Traditional firewalls only protect traffic at the Internet Protocol (IP) level. Web Services effectively tunnel through this layer via standard HTTP(S) and expose the organization's applications to completely new threats. We need to ensure that only valid requests for valid services from genuine clients get inside the ESB.

The new frontier for hackers is systems built on Web Services and XML, seen as inviting due to their popularity but also their immaturity. One class of such attacks is defined as XML threats, where an XML document that has been structured to do harm is sent to the system. If a system can be accessed by outsiders (such as through the Internet), someone could send messages to that system in order to damage it, or even just to consume resources. It is possible that the offending client could even be authorized to use the system, but is trying to exploit that authorization in some inappropriate way.

To cover these security threats we could either define specific security rules to protect our global system, or install and configure specific security solutions to avoid security holes. These solutions could lead us to not using the ESB for certain integrations.

Cost

Before starting to strategically implement an ESB, the investment that goes with it needs to be justified first. An ESB implementation is a trade off between the savings over time as a result of reduced infrastructure complexity, reusability of integration logic, and faster application development, and the cost to initially implement an ESB and maintain it. Typically, the odds of a positive outcome of the justification for an ESB will increase as the amount of SOA-type integration requirements grow. In some cases, the break-even point is just not reached within an acceptable period of time. This could happen if there are hardly any requirements with respect to new integration and interfaces, or it takes a very long time to get the ESB investment paid back.

2.4 Platform choice criteria

Now that we have talked about the different options with regard to implementing an ESB, we are going to consider the functional and non-functional requirements that will help us to choose the platform were the ESB will be deployed. We have already mentioned several aspects in the previous sections, but we now approach the platform decision in more detail.

As stated earlier, the ESB is a key infrastructure component in an SOA. It is the backbone for all communication between service requestors and service providers, and once an ESB is in place, it tends to grow. It will eventually become the component with the highest level of requirements in terms of availability, scalability, security, and performance. When existing mission-critical application functions on System z are going to be accessed through an ESB, this ESB must provide at least the same level of Quality of Service as the application functions accessed. Proximity to data and back-end transactions provides better performance, security, availability, scalability, and integrity.

Important: Even if the ESB and back-end services accessed reside on different LPARs on System z or even different System z machines, we still consider this as "proximate." This is because of the very fast communication techniques between LPARs and between System z machines.

z/OS LPARs and Linux on System z images running on the same box communicate using *HiperSockets* and z/OS LPARs configured in a Parallel Sysplex® communicate using the XCF protocol.

It is important to emphasize that positioning the ESB is not only a question of one platform or another. An ESB is a pattern that could be implemented as a set of different solutions. As we have seen in the pattern descriptions, the ESB even could be spread across multiple platforms. In many large organizations this will be the way to go. Refer to "Requestor on distributed - Federated ESB - Provider on System z" on page 26 for more information.

2.4.1 Functional requirements

From a functional perspective it is important to identify what kind of integration is needed, including what type of services will communicate with each other and which communication protocols and data formats they are expected to use.

Key areas to investigate when selecting an ESB based on functionality are the connectors provided, the standard nodes or primitives provided to perform mediation tasks, programming languages supported to develop custom mediation logic, security, registry support, and tooling.

To a large extent, those functional requirements will lead us to one ESB product or another, but might not be sufficient to make the platform decision. There are cases where functionality is not exactly the same on all platforms, even with the same product. Some connectors and nodes might only be available on certain platforms. In this book we do not focus too much on functionality, but attempt to describe the factors that drive the platform decision, and those are mostly QoS-related considerations.

2.4.2 Non-functional requirements

In the following sections we discuss non-functional requirements (NFRs) and how they influence the platform decision.

Security

Security requirements are sometimes expressed as functional and sometimes as non-functional. It is hard to see the difference. If a security aspect is designed as a function that can be used by an end user it can be regarded as a functional requirement. Security NFRs are security requirements to support the overall solution and provide a secure implementation. Security NFRs many times require measures and technology that an end user usually does not explicitly request. Protection against hacking by implementing firewalls could be seen as an example of an NFR. Assuring that an identity of a transaction in process cannot be changed or manipulated is another example. The end user usually does not bring up these requirements, but “assumes” that the IT infrastructure takes care of this. That’s what makes them NFRs. An example of a functional security requirement is the use of a certain authentication technique to access your bank account.

For the business to remain flexible and responsive, we must be able to give access to our services to different partners as well as to customers, while still requiring the proper authorization to access business services and data. The business must provide access to the data that is required for the business transaction, but also be able to secure other data from unauthorized access. We must prevent rogue data from being replicated throughout the system and to protect the data of the trusted partners. In summary, the business must be open and secure at the same time.

So, what do we need in an ESB from a security point of view and what could influence the platform decision?

To answer this question it is important to understand what the scope of the ESB will be. Will the ESB only interact with intranet service requestors and providers, or also with Internet service requestors and providers? Or, perhaps, only with trusted Business Partner service requestors and providers. If requests come in from an untrusted source, what infrastructure will be in place before those requests hit the ESB? Is there a *Reverse Proxy Security Server (RPSS)* in place?

The aspects to look at are the following:

- When *Secure Socket Layer (SSL)* is used for encryption and mutual authentication between ESB and a service requestor or provider, is there hardware support? If so, is it built-in or through additional processors or cards? Hardware support is important to make the usage of SSL even more secure, but especially to keep acceptable performance.

- How does the platform protect the ESB against hacking attacks? The platform (and not the ESB software) needs to offer protection against all common hacking attempts. Firewalls are a way to do this, but in most cases they are not 100% bulletproof.
- What mechanisms does the platform provide to pass on the security context and credentials of a service request to the back-end service? In many cases it is not acceptable to fulfill this requirement by passing user ID and password in clear inside the message or the message header, but rather at a level that is not visible and cannot be manipulated.

> **Important:** On z/OS, security context of a connection can be kept in the *Access Control Environment Element (ACEE)* of the *Task Control Block (TCB)*. This is a highly secure place to pass on security context and this information cannot be modified by application code. Only authorized functions of the middleware can modify this information. Maliciously implemented application code or a malicious request can therefore not get very far!

- How secure is isolation of data inside the ESB? In one ESB multiple requests from totally different requestors might be active at the same time. Those requests might contain highly confidential data, such as a bank account number, a credit card number, or a social security number. Those requests might even be queued up in the same input queue for the ESB. It is extremely important that the hardware platform and its operating system manage their memory in such a secure way that no "accidents" occur and that malicious application code or a malicious request cannot tap into confidential data of another request.

  This quality depends on the hardware and operating system and is not a matter to be resolved by the ESB alone!
- What built-in reporting mechanisms are available inside the operating system of the platform to feed auditing records?

  Auditing has become a very important requirement and government rules (such as Sarbanes-Oxly) have been established to dictate what should be reported.
- If virtualization and partitioning are used, for instance to run the ESB on one partition and a highly secure database on another partition, how securely are they separated? Again, just a firewall might not be enough, and also firewalls typically do not operate on cross-memory communication or protocols other than TCP/IP.

- How secure is the platform from an operational point of view? Is it physically secured and can only authorized operators access the system? If an operator can access the system, how finely grained will his authorization definitions be? For instance, we might only want to authorize a certain operator to restart an input queue with service requests, but not give that same person the authorization to recycle the entire partition with the ESB on it.

Authentication

Authentication is a security aspect that should be looked at from an end-to-end perspective. It is generally recommended to authenticate users or requests coming from an untrusted zone outside the trusted zone, while the ESB usually runs inside the trusted zone, even if business partners or clients use the ESB. A good practice is to let only users and requests into the trusted zone that have gone through authentication already, and thus the ESB will only have to deal with trusted requests. Nevertheless, a second (weak) authentication might be required at the point of entry in the ESB, in which case the ESB software will need to provide functionality to do this. The quality of authentication is different on different platforms and in certain cases the ESB authentication functions might delegate authentication tasks to the operating system or even the hardware level. On System z it is common that the middleware delegates sensitive functions such as authentication to the O/S. Although the way authentication is performed is typically a functional requirement (user ID/password, certificate, PKI, and so on), the quality of the authentication can be seen as an NFR.

Another point to keep in mind with regard to authentication is that service providers themselves might also want to perform authentication on a request coming from the ESB. In that case it is a prerequisite that the credentials of the request are available from the request. Those credentials can be passed in clear inside the message or message header or in a “hidden” and very difficult to access piece of memory, such as the ACEE segment of the Task Control Block (TCB) in z/OS. The level of security in passing those credentials is an NFR, and the more difficult it is to spoof the credentials, the better this NFR is implemented!

Authorization

Authorization is the process of granting or denying access to a system resource. In the ESB, we must control who has permission to access a certain ESB resource and ensure that users who are supposed to work with this certain resource have the necessary authorization to do so.

Authorization is mostly a functional requirement and gets fulfilled by a combination of functions in the ESB software, the operating system, and eventually other middleware. The way those 3 work together is significantly different between System z (especially z/OS) and distributed platforms, so the way authorization is taken care of can be an argument to choose a certain platform.

It is also of importance what authorization infrastructure is already in place for other services or components. For example, if there is a very mature LDAP and policy server implementation, the ESB might benefit from that implementation. In that case, the platform that best integrates with the existing authorization infrastructure might become the most favorable one for this aspect.

Also, if there are many service providers on System z protected by profiles in RACF®, there might be a requirement for the ESB to use those same profiles as well. This could also influence the platform decision.

Communication security

Depending on the corporate guidelines on protecting network traffic, encryption might be required on requests coming into and going out of the ESB. Because different protocols might be in use, such as HTTP, SOAP over HTTP, JMS, SOAP over JMS, and even specific connector protocols, there might exist an encryption requirement for all protocols being used with the ESB. Encryption support in the ESB solution is an ESB function and it depends on the product which protocols can be encrypted (or not). For encrypted HTTP or SOAP over HTTP requests, support for SSL in the HTTP node of the ESB becomes necessary. For encrypted JMS or SOAP over JMS requests, an MQ channel security exit might become a requirement.

If encryption is only required for the SOAP portion of the request (message level security), the Web Services security has to be used. We discuss this in the next section.

The robustness of the encryption technology used, in part depending on hardware function, can be seen as an NFR, and help determine the best platform to use for this aspect.

Web Services security

New standards and applications for Web Services security are being continually developed and deployed, making it a dynamically changing area. Web Services security as of today can be achieved at two levels:

- Security at the transport level
  Transport security addresses the non-functional requirement that the content exchanged between the ESB and an external partner cannot be viewed by a third party as it travels over the network. Security at the transport level uses the built-in security features of transport technologies such as HTTP and WebSphere MQ. The ESB must support SSL for the HTTP nodes.

- Security at the SOAP or messaging level
  This level is currently being extensively researched and specifications are being developed by groups such as W3C or OASIS. This involves usage of digital signatures, certificates, and so on, at the XML document level. One example is *WS-Security*. Generally speaking, WS-Security is a W3C standard that defines how a Web Services message can be encrypted and decrypted at the SOAP message level to address the requirements of message non-repudiation and prevent messages from being tampered with.

Support for Web Services must primarily come from the ESB software, but encryption functions are ideally delegated to hardware. It is important to make sure that:

- The ESB solution supports an adequate level of the WS-Security framework
- The ESB solution supports delegation of encryption tasks performed in the SOAP nodes to the hardware

Performance

As the intermediator layer connecting different business services, it is critical for the ESB to comply to the performance requirements established by the business applications. Performance is a broad requirement, but for the ESB we are primarily interested in *response time* and *throughput*, in continuous mode and at peak times.

Throughput can be expressed in the amount of requests coming into and going out of the ESB or the amount of mediations being performed per time unit. Different throughput requirements can exist for different types of requests or mediations. Important for achieving the throughput requirements is that no bottlenecks occur before reaching the required throughput numbers. Bottlenecks can occur unexpectedly if a detailed test has not been conducted first. Bottlenecks might be the result of factors outside the scope of the ESB software and therefore it is important that the ESB load, once tuned, must be able to run with the required throughput independently of other workloads on the same system. Resource tuning and assignment to certain workloads, as well as the flexibility to assign more resources dynamically to the ESB workload when needed, are essential to meet the SLAs.

Response time can be expressed as the time needed for a reply on a service request to come back. The time is measured between a certain start point and end point. This requirement gets combined with the throughput requirement because a typical SLA for an ESB will require that both the requested throughput rate and response time be met at the same time. If the workload increases temporarily, maybe even ten- or a hundred-fold, the response time should not collapse because of this.

Performance is determined by several things:

- ESB middleware
  Good ESB software can help a lot in achieving acceptable performance by doing things efficiently, with as little overhead as possible and maximum reuse of memory (caching and so on).
- Operating system
  The operating system provides key functions to the ESB middleware, such as memory management, thread management, access to resources, workload prioritization, communications, and so on. Operating system tuning can help to get the best performance in an ESB, but it is not the biggest performance factor.
- Hardware
  Well-designed hardware also influences ESB performance, but again, it is not the biggest performance factor. Should the hardware be a bottleneck in reaching performance, it is in most cases easy to resolve. Hardware performance includes the processors, I/O channels, communication channels, memory, and so on.
- Mediations deployed
  In most cases this factor has the greatest impact on performance. A mediation will typically become more resource intensive, and therefore exert more influence on performance, as the amount of transformations and the execution of custom logic increases. The programming language in which this custom logic is written is also of importance.
- Request data format
  The more standardized the data format of a request is the less transformation will be required. This helps performance. The ideal would be to use the same data format coming into and going out of the ESB. Protocol conversion is not as much of an issue because whatever protocol is used inbound or outbound, the ESB will need to set up the communication anyway, by means of specific nodes.
- Protocols
  Some protocols are faster than others for the ESB to handle. Obviously HTTPS will require more resource than HTTP. Some specific connectors might use native protocols that might or might not be performance friendly.

Not surprisingly, performance is a complex issue and requirements should be very well described. It is also important to have a clear understanding of what factors actually influence the performance of a solution. The hardware and O/S play a role in performance, but they usually do not make or break your performance requirements. It is the solution being deployed, and especially the way the mediations are designed, that determines your performance for the most part. Too often the focus drifts towards processor speed as the only thing that seems to determine the performance of a solution.

Some testing could be done to identify the performance characteristics of the ESB solution. This testing could include:

- Gradually increasing the number of simulated business service requestors and providers to test for linear scalability and see when bottlenecks start to occur
- Measuring the additional performance cost of different security mechanisms for authentication, data integrity, and confidentiality, and establishing trust between servers
- Comparing the performance cost of accessing a service via different connectors
- Investigating the performance impact of varying message lengths and data formats

Scalability

Being able to meet the performance requirements in a “steady state” workload is one thing, but meeting those same requirements when the workload suddenly increases is another. Nobody really desires to “reserve” resources just for the case when there is suddenly a 100-fold workload increase. Resources should be available for this peak, yes, but reserved, no. Because the resources that we are talking about are ultimately hardware resources managed by the O/S and hardware together, scalability is definitely a platform Quality of Service issue. The way resources can be dynamically increased is a function of the O/S and hardware together. Therefore, we can say that scalability is one of the most important Quality of Service requirements of an ESB implementation.

A deployment of an ESB needs to be able to scale both *vertically* and *horizontally*. It should support both types of scaling since its architecture topology could grow both ways. This could be done by creating runtime topologies that can have multiple O/S images, ESB servers, queue managers, queues, or multiple instances of message flows.

Vertical scaling

Vertical scalability is the ability to increase the existing capacity of the hardware or software by adding resources. Ideally, those resources can be added dynamically and without complex reconfiguration of the system. Examples of vertical scaling include adding real or virtual storage, CPUs, threads, paging space, networking bandwidth, and I/O channels and devices to make the serving component operate faster and with increased capacity.

Important: Vertical scaling should be seen as the least intrusive way of increasing capacity, and there are big differences between different platforms regarding their ability to scale vertically. Some platforms might not scale vertically at all and in those situations the ESB's capacity can only be increased using horizontal scaling.

Ideally, vertical scaling happens dynamically and is based on workload offered to the ESB. We see later in this chapter that z/OS is extremely sophisticated in this matter.

Horizontal scaling

Horizontal scalability is the ability to connect multiple hardware or software entities so that they work as a single, logical unit. Duplicates of components used in the ESB might suddenly start to exist if a sudden increase of workload occurs. Examples are starting additional ESB servers and queue managers on additional system images, or enabling CPUs "on demand." With servers, for example, we can increase the speed or availability of our solution by adding more servers and using clustering and load balancing technologies. The more sophisticated forms of these technologies are sometimes referred to as workload management.

Horizontal scalability through workload management is one of the ways to increase the amount of processing capacity that is available to the ESB. There are several options for sharing the workload between all horizontal nodes, specifically:

1. Hardware technologies provided by the physical platform that emulate the ESB as though it were working in a single node.
2. Software technologies independent of the ESB, with the same solution described in option 1, emulating the ESB as though it were working in a single node.
3. ESB-owned solutions, where the ESB is prepared to have different instances and to share the workload between all the instances deployed.

It is not possible to say which solution is better because this depends on the different products selected. In theory, the hardware solution should have better performance and stability since it is based on hardware technologies.

Important: Although scalability is determined primarily by the O/S and hardware functions, it is important that the ESB middleware exploits those functions. The better the ESB software is integrated with the O/S and the hardware it runs on, the better the performance will be. On the contrary, however, software vendors are generally abstracting the platform as much as possible in their software, and sometimes a software product on one platform is a straight port from another, without any modifications for exploiting platform-specific strengths. IBM’s software on z/OS, including IBM WebSphere Message Broker and IBM WebSphere ESB, are not straight ports and are customized and optimized for z/OS. IBM’s software on z/OS in general is well integrated with z/OS and System z.

Availability

There are several ways to describe the degree of availability that a system requires. The availability of a system can be expressed as a number of nines or by using different terms. The *number of nines* term represents the percentage of time for which the system is available. Three nines means that it is available 99.9% of the time; five nines means that it is available 99.999% of the time. These numbers become much more significant when we look at these figures in terms of downtime over a fixed period. For example, over a period of one year, a system with 99.9% availability would have 8.75 hours of downtime, while a system with 99.999% availability would be down for just 315 seconds.

The following terms are used to describe availability:

- **Continuous availability**
  Describes a system that experiences no discernible downtime, where neither scheduled nor unscheduled outages occur. A continuously available system detects the error and immediately provides an alternative component that is already ready to go. Also, the system should support the scheduling of planned maintenance by allowing workload to be transparently transferred away from the components or subsystems that are the subject of the maintenance activity. Although continuous availability seems difficult to achieve, it is possible to obtain such availability by combining hardware, software, and operational procedures that can mask outages from the user so that the user does not perceive that a system outage occurs.
- **Continuous operation**
  This term describes a system that experiences no discernible downtime due to scheduled outages. However, this system's availability will not be as high as it would be with a continuously available system because it might suffer unplanned outages.
- **High availability (HA)**
  This term describes a system that can detect a single failure and react to it

automatically within a matter of a few minutes at most. This kind of system will operate with some amount of planned and unplanned outages, but with minimum impact. There are two significant aspects in this definition:

  – The system should survive a single failure, but a second failure might result in a loss of service.
  – The detection of a fault and the triggering of an action to recover from it should be automatic, that is, require no manual intervention.

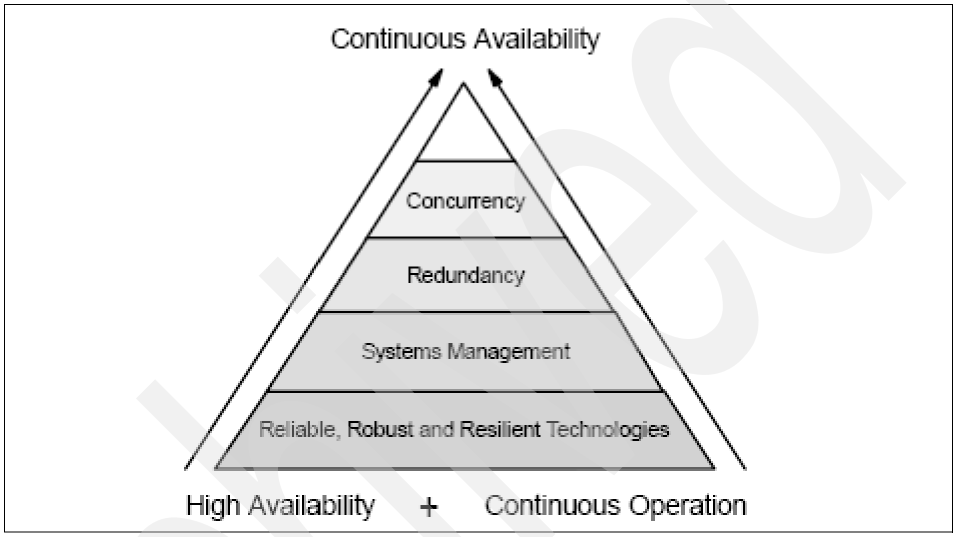


Figure 2-12 System availability definition and references

We have already stated earlier that the ESB will probably become the highest available layer of your infrastructure and will need to offer at least the same availability as your highest available business service connected to this ESB. Individual mediations or certain hubs in the ESB might not need continuous or even high availability, but it is not unlikely that the ESB as a whole will be required to have continuous availability.

Availability of the entire ESB solution as experienced by a service requestor encompasses availability of the following components:

- Hardware
  The hardware that the ESB runs on must technically be available within the requirements set. Hardware includes CPUs, storage, switches, routers, cables, and so on.

  Note: A large-scale ESB might run on a number of different hardware devices, depending on the hardware platform chosen.

- Operating system
  As with the hardware, the O/S must be available within the requirements set.
- The ESB infrastructure itself
  This includes the ESB software, queuing mechanisms, storage used by the ESB, and so on.
- Communication
  TCP/IP or other protocols used to connect to the ESB.
- Connectors and adapters
  Connectors and adapters connect the ESB with runtime containers of the service requestor or provider.
- Deployed artefacts, such as mediations
  The way a mediation is designed can impact availability. Certain unnecessary dependencies in the flow should be avoided. The ability to deploy custom code in the ESB is a good thing, but also introduces a risk of bad code.
- Back-end services
  Of course, if the service provider is not available, the mediation in the ESB will come to a halt and eventually time out, unless there is an alternative service provider available. Having alternative service providers available, especially for unreliable service providers, is important in an HA ESB. With this goes the intelligence that an ESB and the mediations must have to dynamically switch to an alternative service provider, based on its availability.
- IT staff
  Should there be a problem that requires human intervention, IT professionals must be available with adequate skills.
- Software support
  Having excellent support from the ESB software vendor is important in case of severe outages that cannot be resolved by yourself. Because the ESB software might run on a hardware platform from another supplier, use an O/S of yet another supplier, and connect with service providers running in containers of still other suppliers, this matter might get quite complex and result in a situation where you lose time while suppliers are pointing at each other for a resolution. Although many companies try to avoid vendor lock-in, from a software support perspective it is an advantage to have as many components as possible from the same supplier.

Making sure the end-to-end ESB solution meets a certain availability is an IT architect's job and everything must be factored in. Availability of the infrastructure (hardware, O/S, ESB software, communication, service provider runtimes) increases if redundant components are available and the solution automatically switches to those redundant components in case of a failure. Redundant components must exist at the level of CPUs, internal and external storage, communication channels, O/S images, ESB containers, service provider containers, inbound and outbound queues, and so on. Basically, everything that is part of your ESB infrastructure must be looked at and the question must be asked how this redundancy can be achieved and if dynamic switching to alternate components is secured.

As with scalability, the ability to meet supreme availability of the ESB solution depends to a large extent on the platform capabilities. We do not expect the middleware and definitely not the application to provide all of this functionality. Availability is therefore a key platform choice criteria.

Recoverability

Recoverability is the time to come back to a normal status from a failure. This failure can be anything, from running out of memory to a total power outage of the data center. Some types of failure are not major and can be corrected without taking any actions; others can be corrected automatically by the system; and still others need human intervention. Failures might occur in the ESB software, the mediations deployed, or somewhere in the ESB "ecosystem" including the hardware, the O/S, databases used, and so on. It is possible to write a series of books on this topic alone, but we just want to highlight that recoverability is an important platform choice criteria within the ESB scope.

The ideal would be to run the ESB with as much automatic recovery as possible. The more automatic recovery provided by core functions of the hardware and the O/S, the better. A nice way to deal with this requirement is to create a list of all possible failures at the different levels (application, ESB middleware, O/S, hardware, and so on) and describe how recovery would take place for each failure, how quickly, and whether additional software is required for recovery. It is also good to keep an eye on the skills and human staff aspects. Keeping staff on call for eventual recovery actions that could easily be performed automatically by hardware or the O/S is inefficient and very expensive!

Reliability

The need to respond to the business requirements pushes the IT environment to become more and more reliable. Not only is reliability an important part of the functionality of an IT architecture, it is crucial to the operating environment on which the architecture is deployed.

Reliability is also a very broad concept that can be expressed in many different ways and might include many different sub-requirements. We define reliability as the ability to process a service request without data loss or corruption and within the times expected by the user (there is a relation with availability). The overall quality of the Information System and its information has to be safeguarded at all times and the ESB plays an important role in this because most traffic will travel through the ESB. Reliability is many times associated with the aspect of processing transactions using a 2-Phase Commit protocol. Indeed this is one important aspect of reliability, but there are other factors too, such as memory management, data management, and communications. Those factors could jeopardize the quality of the information that travels through the systems if they fail. At a minimum there should always be a confirmation of the service request being received by the ESB and a confirmation of successful execution of the service provider. WebSphere MQ can provide an excellent base for this.

Note: WebSphere MQ is not an ESB by itself, but it can provide the transport layer for all incoming and outgoing messages. The advantage of using WebSphere MQ is that it provides a framework for reliable message delivery.

Transactionality and integrity

Additionally, requirements might arise for transaction processing using 2-Phase Commit in case of multiple data updates.

A *transaction* is used to group operations on data together. If an exception occurs during a transaction each operation performed within that transaction will need to be rolled back. If all operations have been performed successfully, the transaction as a whole needs to be committed. To guarantee the transaction’s and data’s integrity the protocol normally used is the *2-Phase Commit* protocol. Simply stated, we cannot just leave a transaction in the middle with one part of the data updated and the other part not updated and just move on as though nothing happened.

The scope of a transaction could be just the mediation itself, in which case the ESB will need to roll back or compensate the mediation automatically in case of unexpected failures. In this case the situation will be reset up to the point where the message was offered to the ESB.

The scope of the transaction could also include back-end service providers and, more rarely, service requestors. In this case the ESB will not be able anymore to manage the entire end-to-end transaction scope and we will need a *resource coordinator* capable of coordinating the entire transaction spanning the ESB, databases, file systems and transaction managers. This resource coordinator is ideally provided by and integrated with the Operating System. As we see now, reliability has become a platform matter, too, and is an important platform choice criteria.

The platform selected must support all transactionality requirements for the ESB and the mediations designed, and at the same time fulfil other requirements such as availability, scalability, reliability, and performance. This is extremely important when we try to scale a system adding more physical nodes to the ESB solution, since the transactionality or the performance could be affected by distributing the transactions among different nodes.

In order to provide a consistent end-to-end approach to delivery assurance, integrity, and error handling for a chain of service interactions, it will often be necessary to combine several techniques that are used for individual interactions. These techniques might include handling communication failures, the use of synchronous 2-Phase Commit, the ability to handle duplicate messages, and compensation schemes.

WS-AtomicTransaction

In case transactional support is required for transactions making use of Web Services protocols, the following additional WS specifications can be considered.

WS-AtomicTransaction specifies a model for synchronous, 2-Phase Commit of distributed transactions using Web Services protocols. *WS-BusinessActivity* defines an asynchronous model for compensating failed processes using undo actions to reverse the affects of individual steps of the process.

Depending on where the transaction scope starts and ends, support for those specifications needs to be available in the service requestor runtime container, the ESB itself, and the service provider runtime container.

Further information

WS-AtomicTransaction specification:

`http://www.ibm.com/developerworks/library/ws-atomtran/`

WS-BusinessActivity specification:

`http://www.ibm.com/developerworks/webservices/library/ws-busact/`

Transactions in the world of Web Services, part 1 and part 2:

```
http://www.ibm.com/developerworks/webservices/library/ws-wstx1/
http://www.ibm.com/developerworks/webservices/library/ws-wstx2/
```

WS-Coordination specification:

`http://www.ibm.com/developerworks/library/ws-coor/`

Service registry

A growing challenge in an SOA is the management of changes to services already deployed. At a certain point different versions of the same service might exist and different service requestors might work with different versions of the service provider. Even more likely, service interfaces might change quite frequently and service requestors might not be able to stay up-to-date.

A *service registry* can help to address those two issues by offering the ability to keep multiple versions of a service interface at the same time and making the process of using a service interface dynamic.

The role of a service registry is to provide details of services that are available to perform business functions identified within a taxonomy. The service registry is seen as a part of the ESB since it is the central layer through which all services are invoked. However, it quickly becomes apparent that there are much wider uses for a registry that extend far beyond the reach of the ESB. For example, it is valuable to trace, and indeed govern, the life cycle of services from their inception rather than only have knowledge of them once they have reached production. The latter is more a *repository* function and therefore it is not illogical to combine the registry and repository functions into one solution, as is done in IBM's *WebSphere Service Registry and Repository (WSRR)* product.

From the ESB's point of view, a service registry must provide the following capabilities:

- Uploading of service metadata (for example, publishing of services so that other clients can make use of them)
- Access to the registry at development time for data such as service interfaces and endpoints
- Searchable access to the registry at run time for information such as dynamic endpoints and policy
- Enabling the components within the ESB pattern to dynamically respond to changes in the desired use and administration of services

Maintainability

Because the ESB will ultimately have a very high availability and many services will be called over the ESB, it will not be easy to find a time slot to take the entire ESB down for maintenance. If an ESB consists of multiple hubs running on multiple servers, it might be possible to do most maintenance by taking only one hub down at a time. The availability requirements will determine how to design the ESB in such a way that you can keep on doing maintenance tasks.

Maintenance tasks can be required at different levels:

- Hardware
  Sometimes hardware maintenance requires the system to be taken down. If you have extreme availability requirements, you might want to run the ESB on at least two physical machines. In any case, the platform with the least intrusive hardware maintenance would be favored.
- Operating system
  An O/S upgrade usually requires the system (image) to be taken down and

IPLed. To circumvent the impact of this it is a good practice to run the ESB on at least two images, so that one image at a time can be upgraded. The shorter the time and the better automated the O/S upgrade can be made, the better.

Note: If you have done a good job in designing your ESB environment for high availability, taking one system image down will automatically cause all traffic to be routed to other system images.

- ESB software
  An upgrade of a large-scale logical ESB, even implemented across multiple images and even servers, might require the entire ESB to be taken down. This could occur in the case of an urgent fix or a database update in a shared configuration database that affects all ESB hubs at the same time. The impact of this could be circumvented by having at least two ESB implementations that operate independently of each other, especially regarding the usage of configuration data. In that case one ESB can be upgraded at a time, while the other ESB processes all the traffic temporarily.
- ESB configuration database
  Most ESB solutions use a configuration database. Performing maintenance on this database will also effect your ESB. If you have an enormous ESB with many hubs implemented on many servers and system images, but using one shared configuration database, maintenance on this database could bring down your entire ESB. As in the case of maintenance of the ESB software, the impact of database maintenance can be circumvented by having two ESB implementations with physically different configuration databases.

Good planning is a critical part of properly and safely performing maintenance. But platform characteristics with respect to performing maintenance tasks also significantly affect success. For instance, a mechanism of rolling back updates can be very important.

2.5 ESB solutions on System z

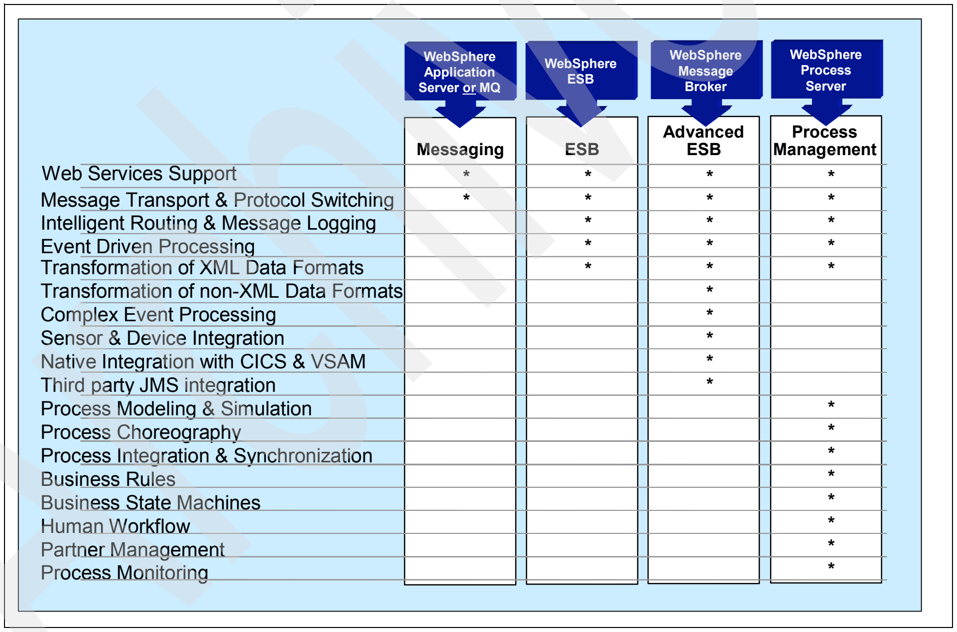
Now that we have identified patterns and requirements related to the ESB and discussed platform selection criteria, it is time to talk about the ESB solutions that are possible currently on System z. For the purpose of this discussion we assume that the ESB will run on System z.

There are various dimensions to consider in those solutions, specifically:

- There is a choice in the operating system to run the ESB on, in particular Linux on System z and z/OS.
- There is a choice in middleware products that can be used in both Linux on System z and z/OS.
- There is even an option to run the ESB “off board,” on the IBM WebSphere DataPower® appliance.

Figure 2-13 shows a list of typical functions in an ESB environment and in which IBM products those functions can be found. For completeness, BPM and workflow functions are also part of this table, but are not typical ESB functions.

> **Important:** The table in Figure 2-13 is valid at the time of writing this book, but the products mentioned might offer more and other functions in the near future. Use this table only to obtain a high-level positioning of the various types of ESBs.



	WebSphere Application Server or MQ	WebSphere ESB	WebSphere Message Broker	WebSphere Process Server
	Messaging	ESB	Advanced ESB	Process Management
Web Services Support	*	*	*	*
Message Transport & Protocol Switching	*	*	*	*
Intelligent Routing & Message Logging		*	*	*
Event Driven Processing		*	*	*
Transformation of XML Data Formats		*	*	*
Transformation of non-XML Data Formats			*	
Complex Event Processing			*	
Sensor & Device Integration			*	
Native Integration with CICS & VSAM			*	
Third party JMS integration			*	
Process Modeling & Simulation				*
Process Choreography				*
Process Integration & Synchronization				*
Business Rules				*
Business State Machines				*
Human Workflow				*
Partner Management				*
Process Monitoring				*

Figure 2-13 High-level ESB types and their functions

Once you have selected the ESB solution that best covers your functional requirements, the next step is to look at the NFRs, discussed in “Non-functional requirements” on page 37.

In the following sections we describe the different ESB solutions and what they offer on System z:

- *WebSphere Application Server* provides an internal bus structure that can be used as an ESB. See 2.5.1, "Using WebSphere Application Server for the ESB on System z" on page 54 for details.
- *WebSphere Enterprise Service Bus (WESB)* is a rich ESB solution based on the WebSphere Application Server infrastructure. See 2.5.2, "Using WebSphere Enterprise Service Bus for the ESB on System z" on page 63 for more details.
- *WebSphere Message Broker (WMB)* is another rich ESB solution based on WebSphere MQ infrastructure. See 2.5.3, "Using WebSphere Message Broker for the ESB on System z" on page 67 for more details.
- *WebSphere DataPower* is an appliance offering security and integration functionality "out of the box." Some of the WebSphere DataPower models provide ESB support. See 2.5.4, "Using IBM WebSphere DataPower for the ESB" on page 81 for more details.
- "Roll Your Own" (RYO) ESB. Many companies have started to build and implement their own framework with ESB functionality. This has been especially popular in J2EE application server environments. Because each RYO ESB is different, we will not discuss this solution in detail.

Attention: We would like to emphasize that in a world of ever-changing standards, maintenance of an RYO ESB demands extensive effort. Also, in the circumstance of a merger or acquisition, one or more RYO ESBs become more of an inhibitor to successful integration of the IT environments than an accelerator.

2.5.1 Using WebSphere Application Server for the ESB on System z

WebSphere Application Server is a suite of transactional servers that implement the J2EE specification. Any enterprise application written to the J2EE specification can be installed and deployed on any of the servers in the WebSphere Application.

A platform-independent topology is shown in Figure 2-14.

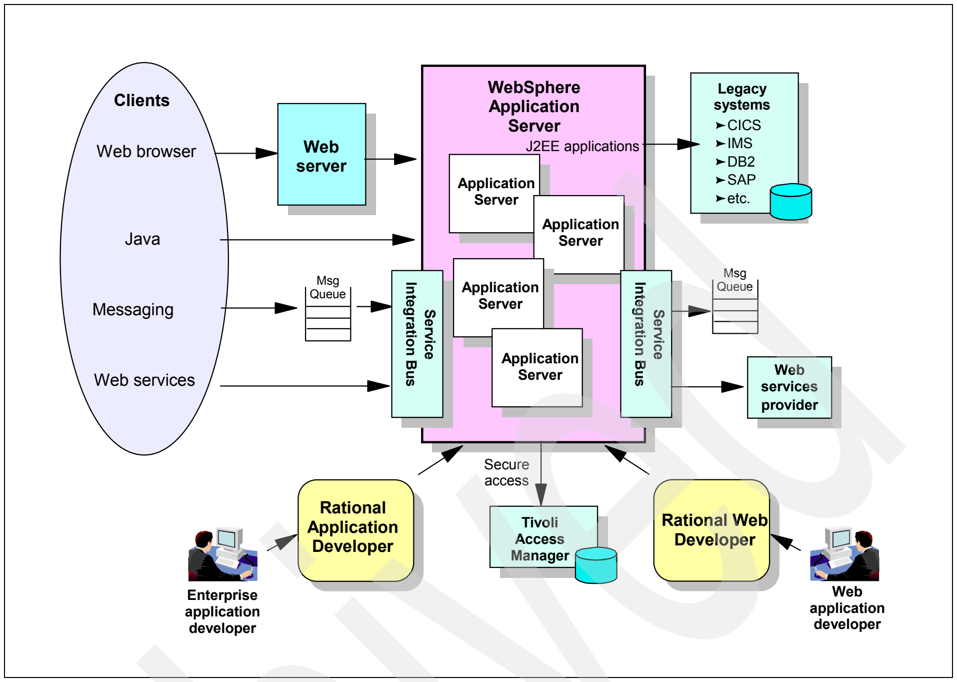


Figure 2-14 WebSphere Application Server topology

On System z WebSphere Application Server is available on both Linux on System z and on z/OS. Both versions offer an internal bus structure, commonly known as the *Service Integration Bus*, or *SIBus*. This bus structure looks very much like a WebSphere MQ network, but also offers mediation capabilities.

Service Integration Bus

The Service Integration Bus in WebSphere Application Server combines support for applications connecting via native JMS, WebSphere MQ JMS, WebSphere MQ, and Web services. It supports the message-oriented middleware and request-response interaction models. As part of this, the SIBus supports multiple message distribution models, reliability options, and transactional messaging.

The functionality provided by the Service Integration Bus includes:

- Multiple messaging patterns and protocols for message-oriented and service-oriented applications
- A J2EE-compliant JMS provider that is the default messaging provider
- The relevant Web services standards that support JAX-RPC APIs
- Reliable message transport capability

- Tightly and loosely coupled communications options
- Intermediary logic (mediations) to intelligently adapt message flow in the network
- Support for clustering to provide scalability and high availability
- Support for the WebSphere Business Integration programming model, which converges functions from workflow, message brokering, collaborations, adaptors, and the application server
- Support for connectivity into a WebSphere MQ network

The SIBus is fully integrated within WebSphere Application Server and on System z it benefits from the QoS provided by the combination of middleware, operating system, and hardware.

Service Integration Bus fulfils the following ESB capabilities as outlined in 2.1.4, “Extended capabilities for an ESB” on page 10:

- Communication
  - Internet routing supported with proxy.
  - Supports SOAP, UDDI, WSDL, and Web services for J2EE.
  - Uses SOAP/JMS channels that support point-to-point and publish/subscribe styles of messaging.
  - Asynchronous messaging supplied through the JMS interface.
  - Synchronous messaging can be managed through Apache SOAP or SOAP/HTTP transports.
- Service interaction
  - WSDL is used as the service interface definition and the service implementation definition.
  - Can publish services to a UDDI directory (either a private or an external directory).
- Integration
  - Java provides the language interface.
  - Protocol transformation support.
- Security
  - Tokens, keys, signatures, and encryption according to the WS-Security specification can be applied to every deployed Web Service.
  - Authentication and authorization is available through the WebSphere Application Server.
  - HTTPS is supported.

  - Proxy authentication can be enabled.
  - Message-level security, as part of the WS-Security specification, is implemented using JAX-RPC.
- Service level
  - Web service performance can be monitored through the Performance Monitoring Infrastructure (PMI), including the number of asynchronous and synchronous requests and responses.
- Message processing
  - JAX-RPC is used as message handlers to provide the ability to extend the basic service implementation. For example, JAX-RPC handlers can be used to manage message-level security (encryption/decryption), logging, and auditing.
- Management and autonomic
- Modeling
- Infrastructure intelligence

The System z difference

The Service Integration Bus is part of WebSphere Application Server. It is possible to install WebSphere Application Server, including SIBus, on either z/OS or Linux on System z.

WebSphere Application Server for z/OS is a specific version of WebSphere Application Server customized to run in an z/OS environment. WebSphere Application Server for Linux on System z is technically equivalent to WebSphere Application Server on distributed platforms. WebSphere Application Server for Linux on System z also is functionally equivalent to that of WebSphere Application Server on z/OS; however, the z/OS version benefits from a wide range of QoS features inherent in z/OS.

Figure 2-15 on page 58 provides a good summary of the differences between the platforms.

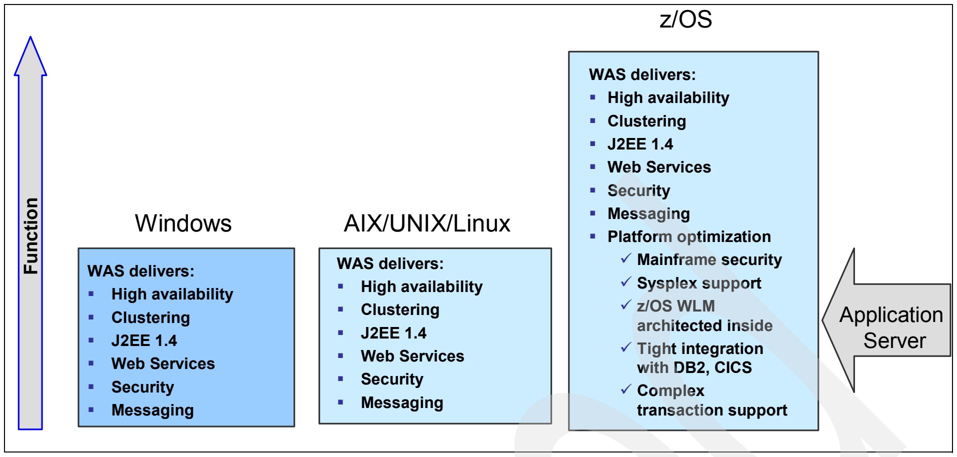


Figure 2-15 WebSphere Application Server on different platforms

z/OS

When implemented on z/OS, WebSphere Application Server provides the benefits of the QoS capabilities offered by z/OS and the underlying System z hardware. They include the following:

- Security

  Besides having access to the security functions available in WebSphere Application Server at the J2EE level, z/OS and hardware security functions can be used, such as:

  – Security administration through z/OS *Resource Access Control Facility (RACF)*

  – Authorization and authentication by means of z/OS RACF

  – Access control and protection by z/OS RACF

  – Encryption support by *Integrated Cryptographic Service Facility (ICSF)* and System z crypto hardware

  – z/OS thread-level security

  Security in WebSphere Application Server for z/OS is a big topic and goes far beyond the security topic on distributed platforms. The best source of information at this moment is the book *Security in WebSphere Application Server V6.1 and J2EE 1.4 on z/OS*, SG24-7384.

- Performance
  WebSphere Application Server on z/OS is a runtime environment optimized for running on z/OS. Although many parts of WebSphere Application Server are common code, parts of WebSphere Application Server requiring z/OS subsystems to operate are customized to z/OS. The extent to which

WebSphere Application Server on z/OS is customized can be seen in Figure 2-16.

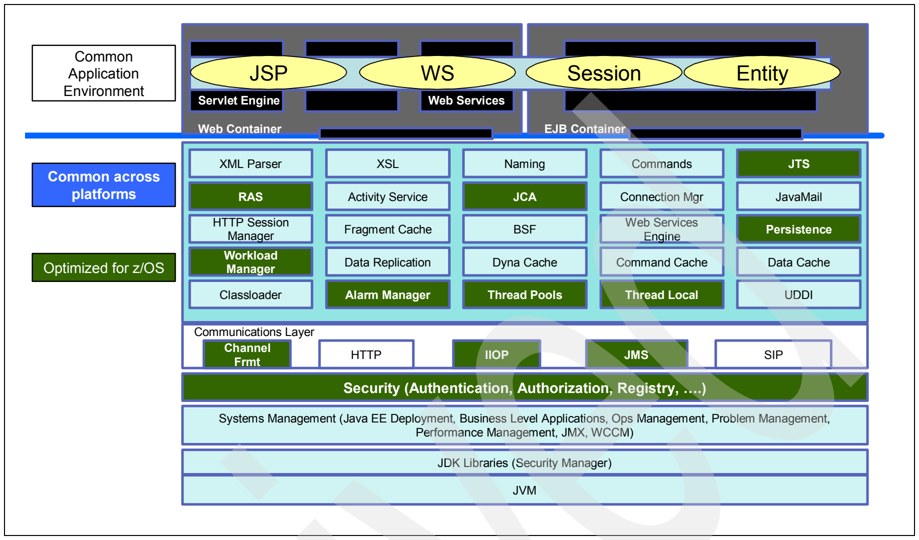


Figure 2-16 WebSphere Application Server common code base

When installed on z/OS, WebSphere Application Server can use local cross-memory connectors to z/OS back-end systems, such as CICS TG for accessing CICS and IMS Connector for Java for accessing IMS. The JVM™ on z/OS has been optimized significantly over the years.

Highly optimized services are used for functions such as security (System Authorization Facility (SAF) callable services) and transactions (Resource Recovery Services (RRS).

- Scalability
  Vertical scalability of WebSphere Application Server on z/OS is mainly achieved by using WLM to distribute workload across WebSphere Application Server Servant regions and starting new Servant regions if necessary. This way more WebSphere Application Server capacity can be mobilized at the server level when there is a peak. At a higher level, WebSphere Application Server servers can be clustered, on the same LPAR, or across LPARs and even machines, to obtain an even wider capacity.

  Figure 2-17 on page 60 shows vertical and horizontal scalability with WebSphere Application Server on z/OS.

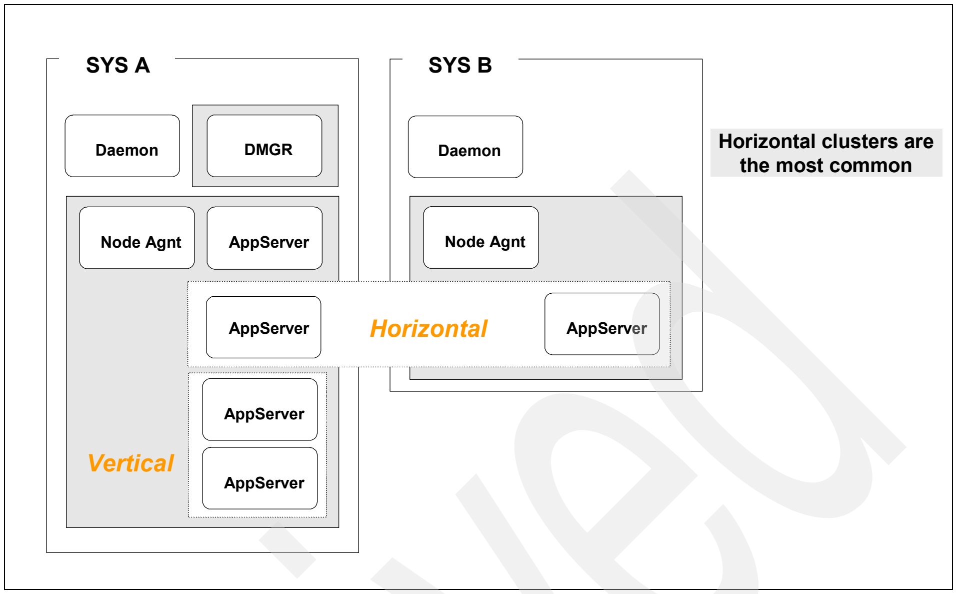


Figure 2-17 Vertical and horizontal clustering in WebSphere Application Server on z/OS

An ideal configuration for maximum scalability is a multi-LPAR configuration in a Parallel Sysplex and WebSphere Application Server servers on multiple LPARs. Sysplex Distributor (SD) and DVIPA can be used to route workload to either LPAR and WLM is used to assign requests to the most available servant region.

- Availability

  An ESB is a critical piece of the enterprise infrastructure that must be continuously available. WebSphere Application Server exploitation of Parallel Sysplex, Automatic Restart Manager (ARM), and WLM features provides a means to assure the ESB is highly available.

  WebSphere Application Server is ARM enabled. Servers registered with ARM can be restarted automatically.

- Recoverability

  Recoverability in WebSphere Application Server on z/OS can be guaranteed by using Parallel Sysplex, eventually in combination with GDPS®. It is important to set requirements with respect to the time it can take to recover from certain failures and to what extent *data loss* or *message loss* can be tolerated in case of a certain failure. If those requirements are very strict, a Parallel Sysplex in combination with GDPS (see "Geographically Dispersed Parallel Sysplex (GDPS) options" on page 440 for the options) will provide the best solution on the market.

Recovering from a failure in any of the LPARs in the primary data center can be achieved using Parallel Sysplex capabilities. Messages in process can be persisted to disk. DB2 Data Sharing can be used for DB2 high availability.

Recovering from a data center failure might require any of the GDPS options discussed in “Geographically Dispersed Parallel Sysplex (GDPS) options” on page 440. Depending on the distance between the primary and secondary data centers and data loss toleration a choice has to be made. By choosing the right GDPS option the workload of an entire WebSphere Application Server infrastructure in the primary data center can be seamlessly taken over by the second data center.

- Transactionality and integrity
  WebSphere Application Server on z/OS supports global transactions using *z/OS Resource Recovery Services (RRS)*. A global transaction in this context includes the activities that take place as part of the mediation running in the SIBus in WebSphere Application Server *and* the interactions between WebSphere Application Server and other resource managers on z/OS. This means that if any of the resource managers (such as WebSphere MQ, DB2, or CICS) used in a mediation fails, the work done in WebSphere Application Server will be recovered. A condition is that connectors and adapters being used support RRS. The concept of running a global transaction with 2-Phase Commit across multiple resource managers (CICS, IMS, DB2, and so on) is a unique z/OS Quality of Service feature.
- Service registry
  WebSphere Application Server has its own Universal Discovery Description and Integration (UDDI) support, but also supports WebSphere Service Registry and Repository (WSRR). This support is part of the core functionality of WebSphere Application Server.
- Maintainability
  WebSphere Application Server on z/OS is tightly integrated with z/OS and benefits from all the typical z/OS maintainability features. New releases and fixes are installed using SMP/E. WebSphere Application Server users can report problems to IBM and solutions to problems become available as PTFs.

  Deploying WebSphere Application Server applications, including SIBus mediations, is done using the WebSphere Application Server Administrative Console, and from this perspective there is no difference with deploying WebSphere Application Server applications to other platforms.

  WebSphere Application Server can be managed and monitored with a variety of systems management tools. See “Availability and performance management” on page 348 for a discussion on tools from IBM.

Linux on System z

WebSphere Application Server on Linux on System z benefits from the System z hardware QoS and the model of running multiple Linux servers on the same hardware.

The hardware benefits offered by System z and Linux on System z are:

- High availability of the hardware platform
- Cryptographic support (PCICA, CPA, PCIXCC, Crypto2)
- Traditional and open I/O subsystems
- High I/O bandwidth capabilities
- Disk (mainframe or SCSI) and tape
- OSA-Express and OSA-Express2 for very high speed communication between z/OS, z/VSE™, z/TPF, and Linux
- HiperSockets for ultra high speed communication between z/OS, z/VSE, and Linux
- Flexibility to run disparate workloads concurrently

However, Linux on System z does not provide all of the features that are available on z/OS. The application isolation in a single Linux image on System z is not as good as that in a single z/OS image. At this time, no WLM equivalent component exists in Linux on System z to handle application workloads, so a single Linux image on System z cannot handle the heterogeneous workloads as well as z/OS.

With respect to fulfilling the NFRs, Linux on System z offers the following:

- Performance
  Superior growth through support of a large number of shared processing engines.
- Proximity of enterprise data stored on z/OS
- HiperSockets, a hardware feature of System z that provides high speed TCP/IP connection between logical partitions
- Scalability
  Fast and easy to clone systems, especially under z/VM®
- Availability
  High Availability via clustering support as with all open systems
- Lower cost hot standby using PR/SM™ support
- Reliability
  Proven high availability of the hardware platform

Conclusions

The SIBus in WebSphere Application Server is a good solution for offering ESB-like capabilities to J2EE applications running in a WebSphere Application Server environment. Services written in J2EE and running in WebSphere Application Server can interact loosely coupled, using open standards such as SOAP and JMS. The SIBus, however, also makes it possible to integrate outside WebSphere Application Server using an MQ Link.

2.5.2 Using WebSphere Enterprise Service Bus for the ESB on System z

WebSphere Enterprise Service Bus (WESB) is designed to provide the core functionality of an ESB for a predominantly Web Services based environment. It is built on WebSphere Application Server, which provides the foundation for the transport layer. WebSphere Enterprise Service Bus adds a mediation layer based on the *Service Component Architecture (SCA)* programming model on top of this foundation to provide intelligent connectivity. If the customer has a lot of Web services in their environment, WebSphere Enterprise Service Bus is likely to be the better product to use.

Because WebSphere Enterprise Service Bus is based upon WebSphere Application Server, it leverages WebSphere Application Server attributes such as clustering, fail over, scalability, security, and a built-in messaging provider. On z/OS, WESB also inherits the QoS provided by the combination of WebSphere Application Server, z/OS, and the hardware.

In terms of functionality, WebSphere Enterprise Service Bus is a super set of the SIBus and adds the following functionality:

- Easy to build mediation layer
  WebSphere Integration Developer (WID) provides a state-of-the-art design and programming environment for mediations deployed to WESB.
- Simplified administration
- Pre-built mediation functions
- Broad connectivity

Mediation functions in WebSphere Enterprise Service Bus are service intermediaries that:

- Operate on interactions between service endpoints (consumer and provider)
- Are administered as part of WebSphere Enterprise Service Bus
- Are created using visual tooling exploiting supplied and custom mediation functions

- Have access to binding-specific header data like SOAP and JMS headers

Mediation handlers in WebSphere Application Server Service Integration Bus are message handlers that:

- Operate on messages traversing the bus
- Are administered as part of the bus
- Are created by implementing Java programs
- Allow access to the full WebSphere messaging header information

Functionality

We divide the functionality offered by WebSphere ESB into three types, which are described in the following subsections.

Messaging routing

WESB manages the flow of messages between SCA-described interaction endpoints and enables the quality of interaction that these components require. Mediation modules within the ESB handle mismatches between requestors and providers, including protocol or interaction-style mismatches and interface mismatches. In an overall SCA-based solution, mediation modules are a type of SCA module that perform a special role, and therefore have slightly different characteristics than other components that operate at the business level.

Mediation

Mediation components operate on messages exchanged between service endpoints. In contrast with regular business application components, they are concerned with the flow of the messages through the infrastructure and not just with the business content of the messages. Rather than performing business functions, they perform routing, transformation, and logging operations on the messages. The information that governs their behavior is often held in headers flowing with the business messages.

Business Objects are the universal data description. They are used as data objects passed between services and are based on the *Service Data Objects (SDO)* standard. In WebSphere Enterprise Service Bus a special type of SDO is introduced, the *Service Message Object (SMO)*.

As a request travels between a service requestor and provider, the logic that controls what happens to it and how it is routed (often known as the mediating or service interaction logic) can be inserted. This mediating logic is performed by mediation modules, which are made up of imports, exports, and flow components that have the following purposes:

- **Import**
  Lets the mediation module reference external services as though they were local services.
- **Export**
  Exposes a mediation module's external interfaces (or access points) to a client that wishes to use a function of the module as a service.
- **Flow components**
  Can carry out the service integration logic, including:
  - Routing
  - Database lookup
  - Database logging
  - Structure transformation

Service endpoint connectivity

WebSphere Enterprise Service Bus supports advanced interactions between service endpoints on three levels, which are as follows:

- Broad connectivity
- A spectrum of interaction models and qualities of interaction
- Mediation capabilities

The product supports diverse messaging interaction models to meet our requirements, including the following models:

- Request-reply
- Distribution models for one-way interactions
- Publish-Subscribe

WebSphere Enterprise Service Bus supports connectivity between endpoints through a variety of protocols and application programming interfaces, such as:

- Java Message Service (JMS) 1.1, provided by WebSphere Application Server, on which WebSphere Enterprise Service Bus is built. Applications can exploit a variety of transports, including TCP/IP, Secure Sockets Layer (SSL), HTTP, and Hypertext Transfer Protocol Secure (HTTPS).
- WebSphere Enterprise Service Bus supports Web services standards that enable applications to make use of Web service capabilities, including:
  - SOAP/HTTP, SOAP/JMS, and WSDL 1.1
  - UDDI 3.0 Service Registry and Web Services Gateway
  - WS-* Standards, including WS-Security and WS-Atomic Transactions

- For back-end applications (such as SAP), several IBM WebSphere Adapters (based on the J2EE Connector architecture (J2C)) are available.

Important: These standards are valid at the time of writing, but might change in future versions of the product.

Because it is built on WebSphere Application Server, WebSphere Enterprise Service Bus can provide smooth interoperability with other products in the WebSphere portfolio, including IBM WebSphere MQ and IBM WebSphere Message Broker. It can use IBM WebSphere Adapter solutions to leverage existing application assets, and also capture and disseminate business events.

Client interfaces included with WebSphere Enterprise Service Bus further extend this connectivity as follows:

- The message clients for C/C++ and Microsoft® .NET enable non-Java applications to connect to WebSphere Enterprise Service Bus using an API similar to the JMS API.
- The Web services client for C++ is similar to the Java API for XML-based Remote Procedure Call (JAX-RPC) and enables users to connect to Web services hosted on WebSphere Application Server from within a C++ environment.

Other features at the connectivity level perform basic protocol conversion between endpoints where the protocol used by the requester to dispatch requests (such as SOAP over HTTP) is different from that of the service provider that handles those requests (such as SOAP over JMS).

The System z difference

Building an ESB that is based entirely on WebSphere Enterprise Service Bus is an option when Web services support is critical and the service provider and consumer environment is predominantly built on open standards.

It is possible to install WebSphere Enterprise Service Bus either on z/OS or Linux on System z.

z/OS

When implemented on z/OS, WESB provides the benefits of the QoS capabilities offered by z/OS and the underlying System z hardware. Those capabilities become available to WESB through the WebSphere Application Server layer. With respect to fulfilling the non-functional requirements, the statements made in “The System z difference” on page 57 also apply to WESB on z/OS.

Linux on System z

WebSphere ESB implemented on Linux on System z benefits from the System z hardware QoS and the model of running multiple Linux servers on the same hardware. With respect to fulfilling the non-functional requirements, the statements made in “Linux on System z” on page 62 also apply to WESB on Linux on System z.

2.5.3 Using WebSphere Message Broker for the ESB on System z

WebSphere Message Broker (WMB) is a powerful solution that allows both business data and information, in the form of messages, to flow between disparate applications and across multiple hardware and software platforms. Business rules can be applied to the data that is flowing through the broker in order to route, store, retrieve, and transform the information.

WebSphere Message Broker is built upon WebSphere MQ and therefore supports the same transports. However, it also extends the capabilities of WebSphere MQ by adding support for other protocols, including real-time Internet, intranet, and multicast endpoints.

WebSphere Message Broker is used for the distribution and routing of messages from disparate business solutions. It can distribute information and data, which is generated by business events in real time, to people, applications, and devices throughout an enterprise. It has support for multiple transport protocols and extends the flow of information in an organization beyond point-to-point, utilizing flexible distribution mechanisms such as publish/subscribe and multicast.

WebSphere Message Broker is not a good place to implement business logic. It is a bad idea to encapsulate business logic inside the message flow. While WebSphere Message Broker has the ability to route and transform a message based on business rules and values, the real, complex business logic should be put into an application server like CICS or IMS transaction manager or WebSphere Application Server.

One of WebSphere Message Broker’s strengths is its powerful capability for parsing different formats of messages coming from different channels or protocols, manipulating the messages, and serializing the messages in different wire formats and protocols. WebSphere Message Broker is flexible enough to support different messaging paradigms, like synchronous and asynchronous behavior; and different message interaction patterns, like fire-and-forget, request-reply, publish-subscribe, and aggregation.

Applications also have much greater flexibility in selecting which messages they want to receive because they can specify a topic filter or a content-based filter, or both, to control the messages that are made available to them.

WebSphere Message Broker provides a framework that supports supplied, basic functions along with user-defined enhancements to enable rapid construction and modification of business processing rules that are applied to messages in the system.

Figure 2-18 on page 68 shows a platform-independent overview of WebSphere Message Broker.

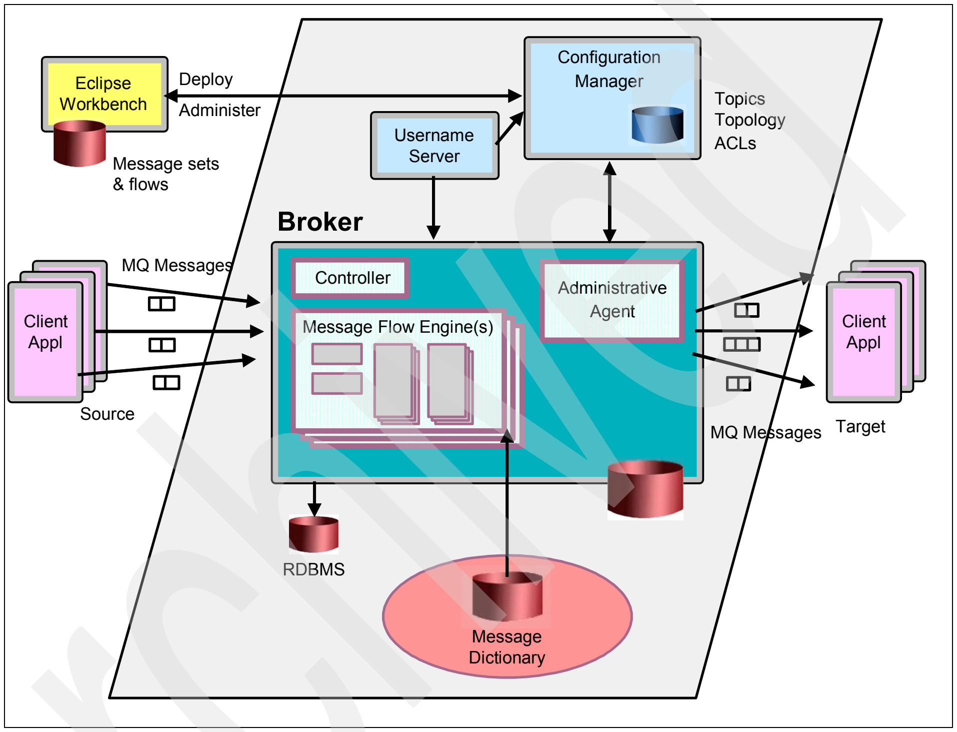


Figure 2-18 WebSphere Message Broker

In this section we discuss the following WMB functionality in more detail:

- Service virtualization
- Transport protocol support and conversion
- Message models and transformation
- Dynamic message routing
- Custom mediation support
- Interaction pattern support
- Integration with other enterprise information systems

- Quality of service (QoS) support
- Service Registry access
- Ease of administration

Service virtualization

Service virtualization is a core functionality of an ESB. The service requestor, in both its application logic and deployment, does not need to be aware of the realization of the service provider. The service requestor does not need to be concerned about the programming language the service provider is written in, its runtime platform, its network address, its communication protocol, or even whether there is a service provider implementation available. The requestor only has the responsibility of connecting to the bus and placing the request.

WebSphere Message Broker uses message flows to accept incoming requests using different formats, protocols, and communication channels. It can determine the destination for the request, convert the request if necessary, and then send the request to the service provider.

The service provider takes the request from WebSphere Message Broker (the bus) in its native format, processes the request, and sends the reply back to the bus, which sends the response to the service requester. The provider does not need to know the origin or format of the request.

Important: Again, we would like to emphasize that there is no hard requirement for any specific protocol to be used in an SOA. One of the key element of success in an SOA is service virtualization, which is primarily an ESB feature and responsibility. A good level of service virtualization can be achieved by using Web Services, but other protocols also can achieve this goal.

Transport protocol support and conversion

The use of an ESB eliminates the need for the service requester and service provider to use the same transportation protocol.

WebSphere Message Broker provides support for a variety of transport protocols for both inbound and outbound connectivity that extends the scope of a common ESB. Protocol switching with WebSphere Message Broker essentially means that a message flow receives a message over one transport protocol and sends it out over another. Protocol switching is automatic, but needs to be configured as part of the design stage of the message flow that will be deployed to the broker.

Depending on the platform that WebSphere Message Broker is running on, there is also support for WBI Adapters. These adapters provide support for other transport protocols, many times specific for a certain application solution. For the latest information about WBI Adapters, use this Web page as a starting point:

```
http://www.ibm.com/software/integration/wbiadapters/wbia/
```

WebSphere Message Broker provides universal connectivity between applications that use a variety of transport protocols. WebSphere Message Broker enables connectivity between applications or business processes that use transport protocols such as SOAP over HTTP, SOAP over JMS, HTTP(S), Java Message Service (JMS), WebSphere MQ, CICS EXCI, TCP/IP, FTP, MQe, MQ telemetry, and user-defined transports. For the latest specifications on protocol support, refer to the following Redbooks publications:

- *WebSphere Message Broker Basics*, SG24-7137
- *Implementing an ESB using IBM WebSphere Message Broker V6 and WebSphere ESB V6 on z/OS*, SG24-7335

Message models and transformation

Much of the business world relies on the exchange of information between applications. The information is contained in messages that have a defined structure that is known and agreed upon by the sender and the receiver.

Applications typically use a combination of messages, including those that are defined by the following structures or standards:

- C and COBOL data structures
- Industry standards such as SWIFT or EDIFACT
- XML DTD or schema

We can model a wide variety of the message formats that can be understood by WebSphere Message Broker message flows. When the message format is known, the broker can parse an incoming message bit stream and convert it into a logical message tree for manipulation by a message flow. After the message has been processed by the message flow, the broker converts the message tree back into a message bit stream.

WebSphere Message Broker typically supplies a range of parsers to parse and write message formats. Some message formats are self-defining and can be parsed without reference to a model. An example of a self-defining message format is XML. In XML the message itself contains metadata as well as data values, enabling an XML parser to understand an XML message even if no model is available. Most message formats, however, are not self-defining. As examples, a binary message originating from a COBOL program and a SWIFT formatted text message do not contain sufficient metadata to enable a parser to understand the message. The parser must have access to a model that describes the message to parse it correctly.

Dynamic message routing

In a service-oriented architecture, an ESB should have the capability to decide the message destination dynamically, thus providing the location transparency of services. With WebSphere Message Broker, the routing can be based on message header or content, or on customer-defined policy.

WebSphere Message Broker can access the message header attributes at run time, making it possible to carry the routing information in the message header like MQRFH2 or MQMD, and route the message based on the header information.

It is worth mentioning here that in WebSphere Message Broker V6, one of the significant ESQL enhancements is the new data types: ROW and SHARED. These are very useful if the solution is designed to route the message to different destinations by looking up a rule table. Usually our approach is to save the rule table in a database so that it is accessible to all the message flows. However, a better alternative in Version 6 is to use the SHARED ROW variable. We can set up the lookup table in memory, where it can be shared by all threads of the message flow instances. Memory access also avoids the overhead of accessing a database for each incoming message.

Custom mediation support

WebSphere Message Broker supports customized mediation capability through the use of the following built-in or SupportPac nodes:

- Database nodes
  The database nodes can be used to enrich messages at run time from an ODBC data source or to perform operations on database data. These nodes provide a flexible interface with a wide range of functions. We can specify transaction behavior in the message flow, specifying whether changes are committed immediately or after the entire message flow is successful.
- File and data set related nodes
  WebSphere Message Broker provides a set of SupportPac nodes that can be used to enrich messages or used as logging facilities for monitoring purposes.
- Other supplied nodes
  An LDAP plug-in node provided by SupportPac IA08 can be used to perform an LDAP lookup using certain parameters in the message, and then to enrich the message using the result from the LDAP server. We can also use the sendmail node provided by SupportPac IA07 to construct and send e-mail using the content in an XML message.

- Custom extensions
  WebSphere Message Broker provides extensive APIs to allow users to write custom extensions to perform customer mediation and message model handling.

Interaction pattern support

When two or more participants interact with others, they always fall into one or more of these *interaction patterns*:

- One-way interaction
- Request-response
- Aggregation
- Publish/subscribe (pub/sub)

Integration with other enterprise information systems

An ESB should have the capability to interact with the existing enterprise information systems within the enterprise. There are many adapters available for interaction with other components within the environment, in particular off the shelf applications like Siebel®, Oracle®, SAP, and also IBM enterprise environments, for example, CICS Transaction Server and IMS.

WebSphere Message Broker supports integration with WebSphere Business Integration Adapters (JMS based) to interact with existing enterprise information systems.

Quality of service (QoS) support

From an architecture point of view, an ESB should only provide capability to support service virtualization, but there are always a lot of Quality of Service issues affecting an ESB. Some important QoS aspects relevant to an ESB are:

- Authentication and authorization
- Non-repudiation and confidentiality
- Transactions
- Various assured delivery paradigms
- Performance and throughput
- High availability
- Logging, metering, and monitoring
- Integration to systems management and administration tooling

Figure 2-19 on page 73 shows an example of WebSphere Message broker in an HA configuration on z/OS.

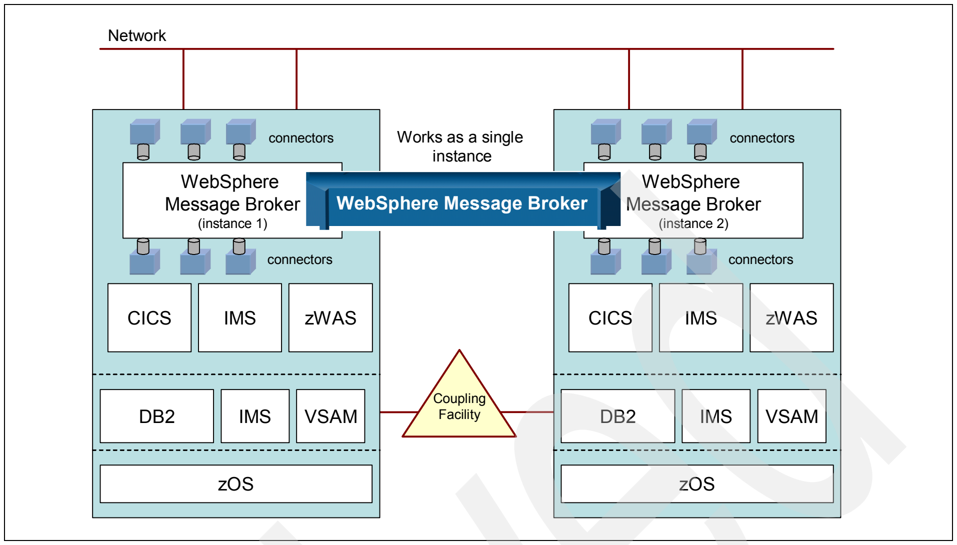


Figure 2-19 Example of WebSphere Message Broker high availability on z/OS

Service Registry access

As the number of services in SOA infrastructures increases, it becomes increasingly important to have an SOA with the capability to:

- ► Find and publish services
- ► Manage service life cycle
- ► Support policy-driven service interactions
- ► Perform change notification
- ► Offer a central service metadata repository

A Service Registry manages the service metadata, enabling selection, invocation, management, governance, and reuse of services leading to a successful SOA. As the service backbone in an SOA architecture, the ESB should have the full ability to access the Service Registry in the enterprise.

SupportPac IA9L is provided to enable WebSphere Message Broker to access WebSphere Service Registry and Repository from within the message flow, thus providing service location transparency capability and flexible service deployment and management.

Ease of administration

An ESB should be easy to manage. WebSphere Message Broker includes the Message Brokers Toolkit, an administration tool that provides both development and runtime administration capabilities.

Components

There are four components in WebSphere Message Broker V6:

- Toolkit
  - In the *Development Perspective*, Message Broker V6 provides user-friendly, Eclipse-based tools that help in developing broker artifacts such as message flows and message sets. Message Flows, message definitions, and any other associated files are packaged into deployment containers called *broker archive (.bar)* files ready for deployment to the broker runtime.
  - An *Administration Perspective* lets operations staff deploy .bar files to any broker within the administrative domain. In addition, the full operational state of each broker is visible and can be controlled from this perspective. The Administrative Perspective uses the Configuration Manager to effect changes and report status. The Eclipse toolkit runs on Linux and Windows. Each developer requires a broker toolkit with a Development Perspective, and each administrator might use a broker toolkit for graphical administration as required. A full suite of command-line tools and programming interfaces is also available for operations management.
  - The Eclipse toolkit runs on Linux and Windows. Each developer requires a broker toolkit with a Development Perspective, and each administrator might use a broker toolkit for graphical administration as required. A full suite of command-line tools and programming interfaces is also available for operations management.
- Broker
  - The *Broker* is the run time where deployed flows execute within containers called execution groups, which appear as z/OS address spaces. Execution groups provide an excellent opportunity for vertical scaling and isolation.
  - Operational personnel have a full range of commands, both JCL and console-based, that let them view and control the operational state of a broker.
- Configuration Manager
  - The *Configuration Manager* is the single management and administration component for a collection of brokers in a domain. The configuration manager controls all operational actions via access control lists. It also

monitors broker state and holds topology information related to deployments and inter-broker connectivity.

– All users' toolkits are connected to a configuration manager.

► User Name Server

– The *User Name Server* component is used in pub/sub networks to determine the set of users and groups used for pub/sub security checking. This set can be either the users defined to the z/OS operating system, or those defined with a user-created program or file. These values are sent to both the configuration manager and the broker for subsequent administrative and runtime processing.

A typical configuration is shown in Figure 2-20. The diagram represents a broker domain containing three brokers under the administrative control of the Configuration Manager. We can perform development and administrative tasks using the Message Broker's toolkit (in this configuration, a user name server is not shown.) The Message Broker runtime component is responsible for routing and transforming messages between applications using message flows, nodes, and parsers.

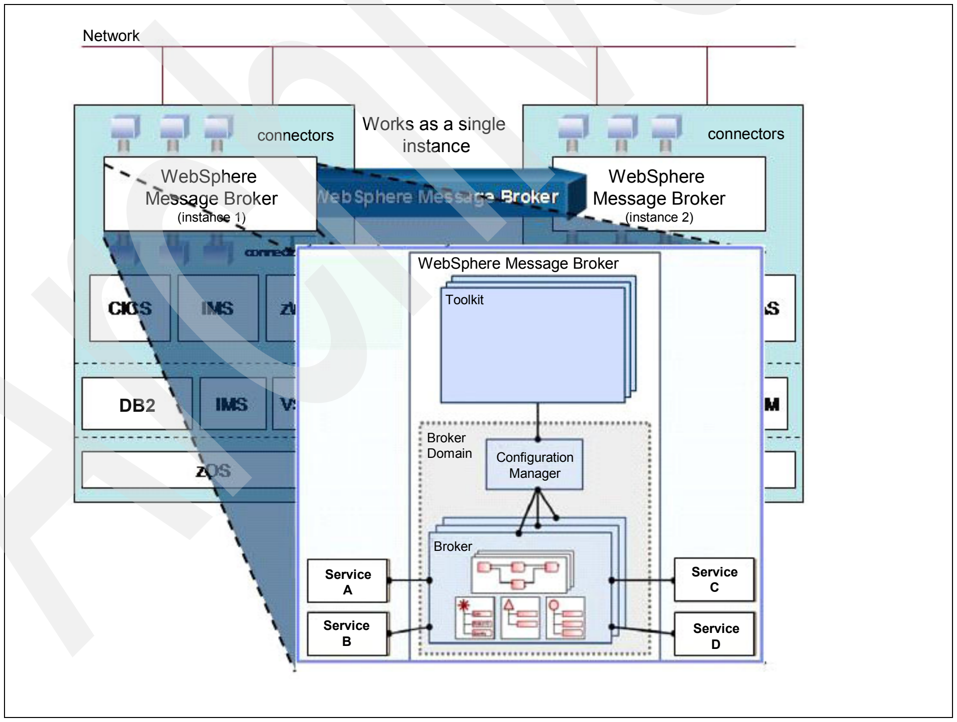


Figure 2-20 WebSphere Message Broker components

The System z difference

It is possible to install WebSphere Message Broker either on z/OS or Linux on System z.

WebSphere Message Broker for z/OS is a specific version of WebSphere Message Broker customized to run in a z/OS environment. There are both functional and non-functional differences between WebSphere Message Broker on z/OS and WebSphere Message Broker on distributed. WebSphere Message Broker for Linux on System z is functionally equivalent with distributed, but does benefit from certain QoS aspects of the System z hardware.

z/OS

When implemented on z/OS, WMB provides the benefits of the QoS capabilities offered by z/OS and the underlying System z hardware. They include the following:

- Security
  - Besides having access to the security functions available in WMB, z/OS and hardware security functions can be used, including:
    - Security administration through z/OS' *Resource Access Control Facility (RACF)*
    - Authorization and authentication by means of z/OS' RACF
    - access control and protection by z/OS' RACF
  - Execution groups on z/OS run in separate address space and can thus be isolated from a security perspective.
  - z/OS security for WebSphere MQ queues applies.
- Performance
  WebSphere Message Broker is a native runtime environment, optimized for running on z/OS. WMB on z/OS is *not* a ported application from UNIX or Windows, but a unique runtime in the spirit of z/OS. Highly optimized services are used for functions such as security (SAF callable services) and transactions (Resource Recovery Services (RRS)). Optimized connectors are available to connect to local z/OS resource managers, such as DB2 (Call Level Interface (CLI) and External CICS Interface (EXCI)). The underlying WebSphere MQ integration on z/OS is also optimized. Performance of flows in WMB is very much influenced by the amount of XML parsing and transformation taking place. Although WMB on z/OS is a very efficient and optimized runtime for performing ESB tasks, performance can degrade significantly by over-engineering flows.
- Scalability
  Workload management is also an important aspect of any ESB platform. You can use the z/OS Workload Manager (WLM) with WebSphere Message

Broker to make sure resources are available for the most important work entering the system. WebSphere Message Broker partitions processing activities into Execution Group address spaces. You can assign these address spaces to WLM service classes so that WLM can control Message Broker processing based on goals established for the specified service class. This practice ensures that high priority requests are handled at the expense of less important work.

Using WLM in a good manner, in combination with Parallel Sysplex, can provide a practically linear scalable environment.

- Availability
  An ESB is a critical piece of the enterprise infrastructure that must be continuously available. WebSphere Message Broker exploitation of Parallel Sysplex, Automatic Restart Manager (ARM), and WMQ Clustering features provides a means to assure the ESB is highly available.

  WebSphere Message Broker relies on the Sysplex-aware resource managers WebSphere MQ and DB2. You can make the queues and tables used by WebSphere Message Broker available to any image in the Sysplex by using *WebSphere MQ Shared Queues* and *DB2 Data Sharing* respectively. The Brokers that participate in a WMQ Queue Sharing Group can continue to process messages on any remaining z/OS image that has a Queue Manager in the Queue Sharing Group. Because of the shared resources, no message destined for a Broker can become trapped in a local resource manager, unlike what would happen with “shared nothing” configurations.

  You can enable WebSphere Message Broker to take advantage of *WebSphere MQ Clustering* in environments that are not Sysplex enabled. The WMQ Cluster assures that any newly arrived messages will be delivered to a remaining Broker when any Broker in the cluster fails. Even though messages that arrived at the Broker prior to a failure might be marooned, this “shared nothing” configuration coupled with ARM support is a substantial availability improvement in the absence of SYSPLEX support.

  WebSphere Message Broker is *Automatic Restart Manager (ARM)* enabled. Brokers registered with ARM can be restarted automatically on either the same z/OS image or another available image.

- Recoverability
  Recoverability in a WebSphere Message Broker on z/OS can be guaranteed by using Parallel Sysplex, eventually in combination with GDPS. It is important to set requirements with respect to the time it might take to recover from certain failures and to what extent *data loss* or *message loss* can be tolerated in case of a certain failure. If those requirements are very strict, a Parallel Sysplex in combination with GDPS will provide the best solution on the market.

Recovering from a failure in any of the LPARs in the primary data center can be achieved using Parallel Sysplex capabilities. Messages in process can then be kept in WMQ Shared Queues in the Coupling Facility (CF). Those messages are persisted on disk and if this is not enough, a second CF can be used as a backup. Any data being used can be stored in a DB2 Data Sharing Group (DSG), accessible from any LPAR in the Sysplex.

Recovering from a data center failure might require any of the GDPS options discussed in “Geographically Dispersed Parallel Sysplex (GDPS) options” on page 440. Depending on the distance between primary and secondary data centers and data loss toleration a choice has to be made. By choosing the right GDPS option, the workload of an entire WMB infrastructure in the primary data center can be seamlessly taken over by the second data center.

- Transactionality and integrity
  WebSphere Message Broker on z/OS supports global transactions using *z/OS Resource Recovery Services (RRS)*. A global transaction in this context includes the activities that take place as part of the message flow *and* the interactions between WebSphere Message Broker with other resource managers on z/OS. This means that if any of the resource managers (such as WebSphere MQ, DB2, or CICS) used in a flow fails, the entire flow will be recovered. A condition is that connectors and adapters are being used that support RRS. The concept of running a global transaction with 2-Phase Commit across multiple resource managers (CICS, IMS, DB2, and so on) is a unique z/OS Quality of Service.

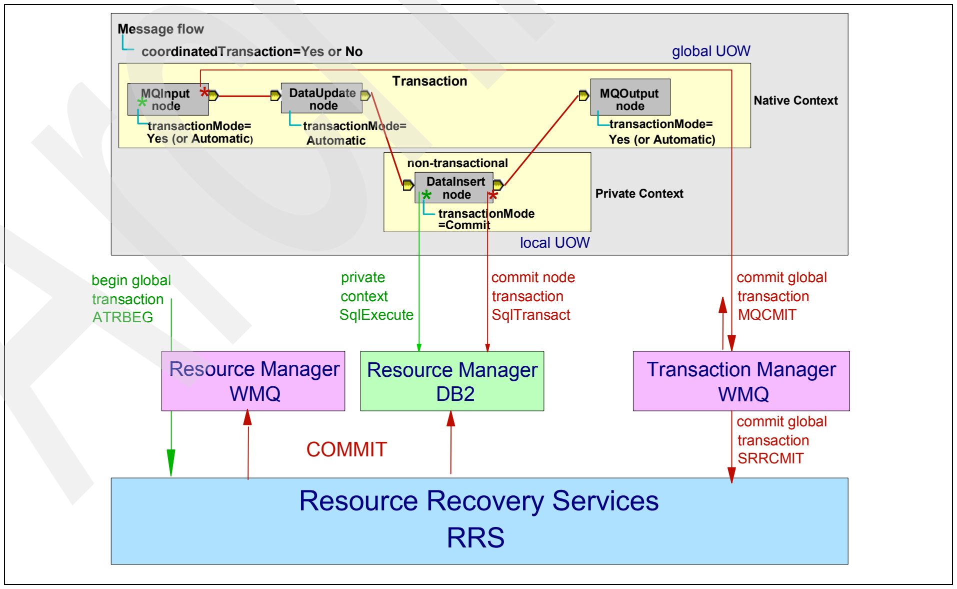

Message flow
coordinatedTransaction=Yes or No
global UOW
Transaction
MQInput node
transactionMode= Yes (or Automatic)
DataUpdate node
transactionMode= Automatic
MQOutput node
transactionMode= Yes (or Automatic)
Native Context
non-transactional
DataInsert node
transactionMode =Commit
Private Context
local UOW
begin global transaction ATRBEG
private context SqlExecute
commit node transaction SqlTransact
commit global transaction MQCMIT
Resource Manager WMQ
Resource Manager DB2
Transaction Manager WMQ
commit global transaction SRRCMIT
COMMIT
Resource Recovery Services RRS

Figure 2-21 Running global transactions in WebSphere Message Broker on z/OS

- Service registry
WebSphere Message Broker on z/OS supports WebSphere Service Registry and Repository (WSRR). This support has been introduced in Version 6.0 as a SupportPac and is integrated further in Version 6.1. WSRR can run on z/OS or somewhere else. Specific nodes are available that can be used to design access to WSRR from flow. Typically, WSRR is accessed just before an outbound service call will be made to an external Web service.

  Figure 2-22 shows an example of a flow with an “SRRetrieveITService” node. This node retrieves the endpoint of a service in CICS that is called in node “CUSTINQ_WebSvc.”

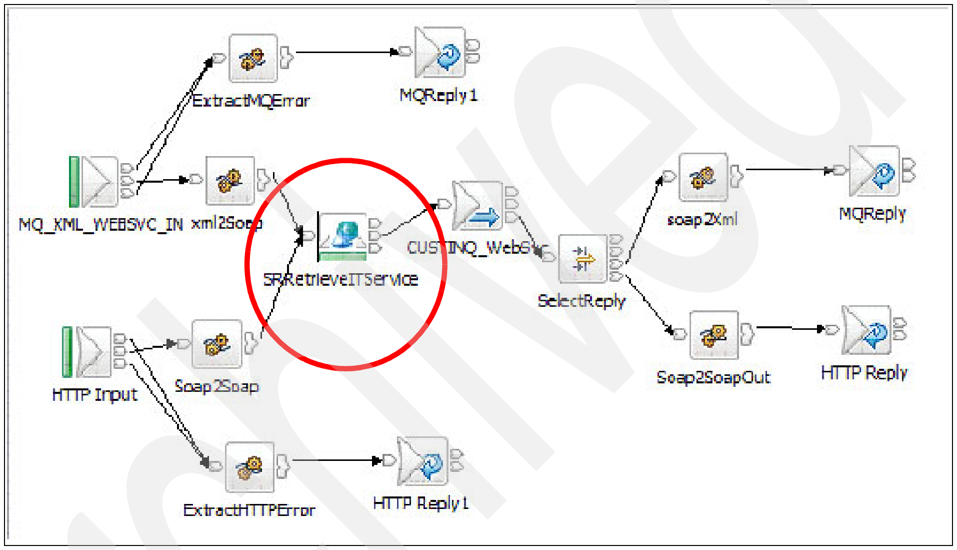


Figure 2-22 Example of an WMB flow including a query in WSRR

- Maintainability
WebSphere Message Broker on z/OS is tightly integrated with z/OS and benefits from all the typical z/OS maintainability features. New releases and fixes are installed using SMP/E. Add-on functions can be installed as WMB SupportPacs. WMB users can report problems to IBM and solutions to problems become available as PTFs.

  Errors in message flows are easily visible in the job output of the Broker address spaces and if this is not enough, tracing can be activated.

  For deploying Broker applications, the WebSphere Message Broker Toolkit Administration perspective is used, and from this perspective there is no difference with deploying Broker applications to other platforms.

WMB and WMQ can be managed and monitored with a variety of systems management tools. See “Availability and performance management” on page 348 for a discussion on tools from IBM.

The runtime of WebSphere Message Broker on z/OS is shown in Figure 2-23.

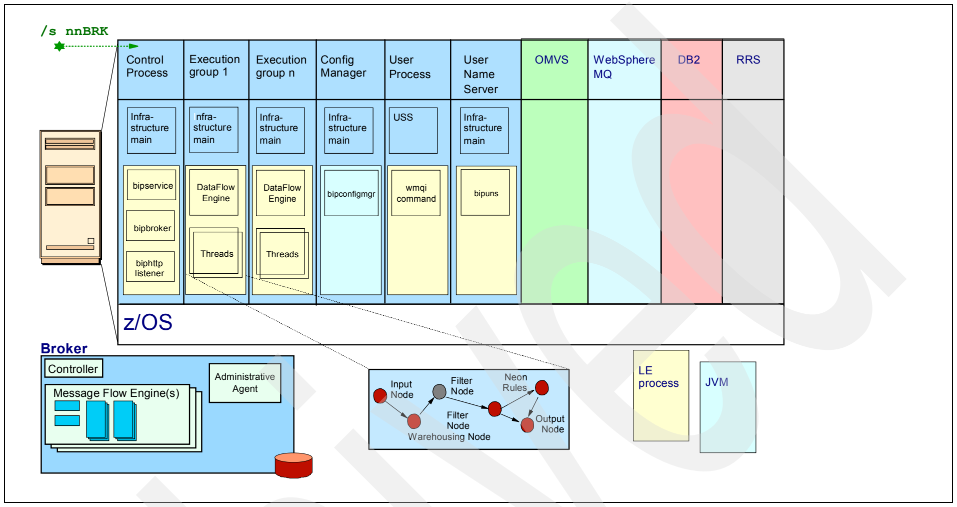


Figure 2-23 WebSphere Message Broker on z/OS runtime

Linux on System z

WebSphere Message Broker on Linux on System z benefits from the System z hardware QoS and the model of running multiple Linux servers on the same hardware.

With respect to fulfilling the non-functional requirements the statements made about WebSphere Enterprise Server Bus in “Linux on System z” on page 62 also apply to WMB on Linux on System z.

Conclusions

WebSphere Message Broker is suitable where advanced ESB functionality is required. WebSphere Message Broker is an option when Web services support is needed and QoS requirements demand the use of mature middleware. WebSphere Message Broker can support the majority of the ESB capabilities that WebSphere Enterprise Service Bus does, but WebSphere Message Broker is not limited to open standards.

2.5.4 Using IBM WebSphere DataPower for the ESB

In this section we discuss IBM WebSphere DataPower, a very popular "appliance" solution for the ESB.

IBM WebSphere DataPower overview

While XML and SOAP enable rich application communication, their emergence in this space has been riddled by three key challenges:

- The size and complexity of traditional software-based middleware product installations has increased installation and maintenance costs of service-oriented architecture deployments and decreased overall solution consumability.
- The text-based, application-centric parsing and processing demands placed on middleware infrastructures have negatively affected overall system performance. While most middleware solutions "scale out," the overall effect is increased license and operational costs as more instances of individual middleware components must be deployed to meet service consumer demand.
- The new genre of application-awareness has likewise led to a new genre of security attacks and exposures that use the architecture of middleware components.

WebSphere DataPower is a hardware and software offering that provides a number of significant functions:

- XML acceleration
- Security enforcement
- ESB functionality

IBM WebSphere DataPower has several important characteristics:

- Optimized hardware, firmware, and imbedded operating system.
- A high level of assurance that the configuration is locked-down.
- Reduced security vulnerabilities.
- Hardware storage of encryption keys and a locked audit log.
- No hard disks, CD ROMs, or USB ports.
- A tamper-proof case that renders the machine unusable if opened.
- Reduced operational complexity because it is truly an SOA appliance. DataPower provides the fastest entry into the SOA world, while at the same time providing an enhanced security environment.

The following typical situation requirements are well suited to a WebSphere DataPower solution:

- Security Gateway
- XML firewall, parsing and validation
- Basic routing
- Content-based routing
- Protocol bridging (HTTP, WebSphere MQ clients, FTP, ODBC, and so forth)

Figure 2-24 gives an overview of these capabilities.

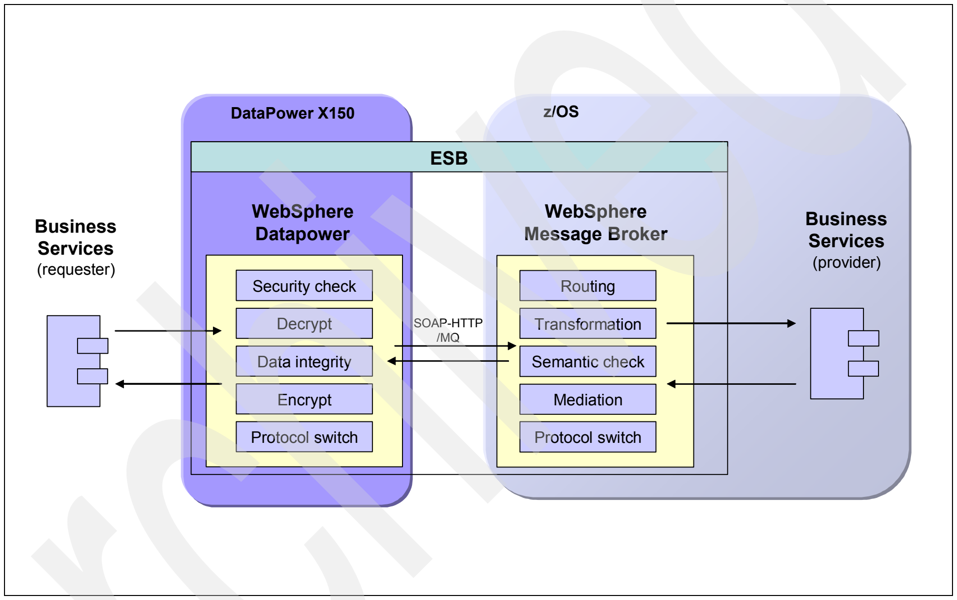


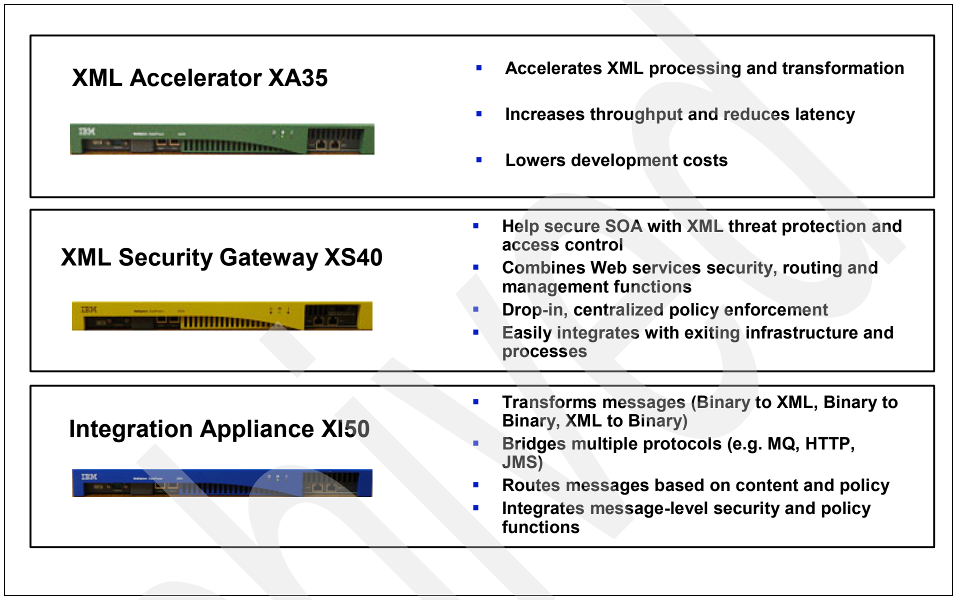
Figure 2-24 WebSphere DataPower in and ESB solution

IBM WebSphere DataPower products

The product line consists of three appliances. The higher model numbers are a functional super set of lower-numbered appliances. When examining the model numbers, it might help to keep in mind that “A” stands for *Acceleration*, “S” for *Security*, and “I” for *Integration*.

All appliances share a basic common engine as well as management and monitoring interfaces. The appliances also support a SOAP (WS-) management interface that enables programmatic access to the appliance's configuration. An Eclipse plug-in, included with the appliances, leverages this interface and provides IDE support. In addition, all appliances come with vast support for monitoring, event logging, and alerting. Supporting a wide variety of event levels and criteria (IP, TCP, HTTP, XML, SOAP), output can be directed to a number of targets including file systems, SNMP, SOAP endpoints, Common Base Event, the console, or a system cache.

The three types of WebSphere DataPower SOA Appliances are shown in Figure 2-25.



XML Accelerator XA35

- Accelerates XML processing and transformation
- Increases throughput and reduces latency
- Lowers development costs

XML Security Gateway XS40

- Help secure SOA with XML threat protection and access control
- Combines Web services security, routing and management functions
- Drop-in, centralized policy enforcement
- Easily integrates with exiting infrastructure and processes

Integration Appliance XI50

- Transforms messages (Binary to XML, Binary to Binary, XML to Binary)
- Bridges multiple protocols (e.g. MQ, HTTP, JMS)
- Routes messages based on content and policy
- Integrates message-level security and policy functions

Figure 2-25 IBM WebSphere DataPower product line

IBM WebSphere DataPower XML Accelerator XA35

The most basic appliance, the XA35, delivers a number of routing and transformation features. It supports routing based on IP, TCP, HTTP, XML, and SOAP criteria. Routing tables can be defined (or queried) to determine the appropriate routing action. Load balancing algorithms such as least-used and round-robin are also available. Additionally, the XA35 can throttle or adjust request rate based on routing criteria to employ traffic shaping. From a transformation perspective, the XA35 provides basic XSLT processing. Built on the underlying premise of all appliances, special-purposed hardware converts one XML schema to another at wire speed. XSLT 1.0 and XPath 1.0 support (with some 2.0 support) is available. The appliance is able to use and apply internal as well as external schema.

IBM WebSphere DataPower XML Security Gateway XS40

The DataPower model XS40 offers security capabilities, such as XML encryption and XML digital signature processing, in addition to its Accelerator capabilities.

Supporting WSS 1.0, WS-Trust, and WS-SecureConversation, the XS40 is the foundation upon which Web Services security can be deployed in an enterprise. Support exists for authorization (access) checking to off board entities such as the Tivoli® Access Manager, Tivoli Federated Identity Manager, RSA, Netegrity, Oblix, and more. There is partial SAML 2.0 support as well as the ability to extract and use security credentials from LDAP, RADIUS, and XKMS. But perhaps the most significant security feature of the XS40 is its firewalling capabilities. In addition to filtering input traffic (as is done in the XA35 routing capabilities), the XS40's firewall enables XML Denial of Service protection. Used in conjunction with schema validation and well-formedness checking, the XS40 is an integral part of an enterprise's network security.

IBM WebSphere DataPower Integration Appliance XI50

The XI50 adds integration capabilities to its accelerator and security abilities. It enables a key concept of *any-to-any* transformation, by which data can be received in any format over any protocol and be converted to any other format over any other protocol using DataPower core high performance transformation technology. That is, in DataPower everything is a transformation. Supported protocols include HTTP, HTTPS, and MQ. Formats include, of course, SOAP/XML, as well as EDI, CICS Cobol Copybook, Corba, ISO 8583, CSV, ASN.1, ebXML, and more.

IBM WebSphere DataPower as an ESB

There can be one or many technology components in any one ESB architecture. There can be many ESBs in an enterprise. The justification for how an ESB is constructed might be based on several constraints, such as legacy requirements, left over infrastructure from acquisitions, or old integration technology not yet sunsetted due to ROI considerations. Regardless, WebSPhere DataPower brings many attributes to an ESB to provide added value to the enterprise.

Of note, DataPower can incrementally provide the following capabilities to an enterprise ESB:

- Enhanced throughput, reduced latency for XML processing, and security processing.
- The very nature of XML and Web Services (for example, SOAP) present a number of architectural difficulties that ESB products must overcome. While XML provides many advantages over other wire-encoding schemes (its ubiquity and simplicity, for example), it is fundamentally more cumbersome to deal with. Parsing XML is tedious and rapidly becomes a performance bottleneck. Transforming XML from one schema to another is challenging. Similarly, understanding and processing SOAP envelopes requires complex XML processing and is generally accompanied by a tremendous amount of

object de-serialization, the creation of in-memory objects that ESB products can understand.

- WebSphere DataPower appliances are targeted toward specific functional processing focused on XSL processing. DataPower's hardened appliance architecture has the potential to achieve higher performance than traditional general software stacks sitting atop general hardware, in the domain of its core primary mediation services. It leverages purpose-built software, firmware, and hardware to build a synergistic pipelined system centered around processing message packets based on open standards.
- Securing the edge of the ESB via XML firewall and XML threat protection (eligible for DMZ deployment).
- Higher level of security assurance including DoS protection.
- Advanced Web Services gateway functions.
- Remote Device management (CLI, signed and encrypted logging, and so forth).
- Enhanced WS-* (in particular WS-Security support).
- Web services gateway functionality (eligible for DMZ deployment).
- Optimized any-to-any transformation, including non-XML via WTX.
- Other Integration client support.
- WebSphere MQ.
- Database ODBC Interface.
- TIBCO EMS.
- IMS Connect.

Figure 2-26 shows how WebSphere DataPower can act as an ESB or Web services gateway to access IMS, CICS and DB2 on z/OS.

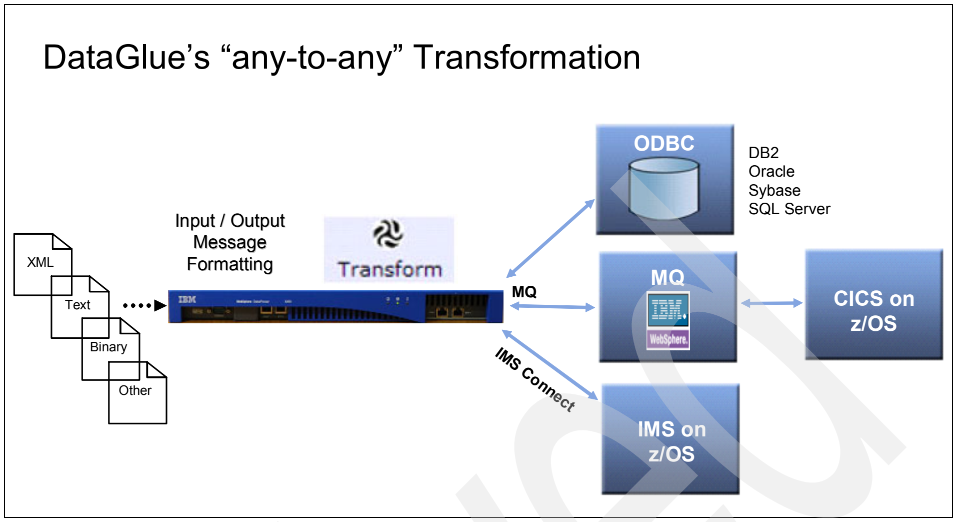


Figure 2-26 Accessing CICS and IMS applications on z/OS through WebSphere DataPower

IBM WebSphere DataPower and System z

IBM WebSphere DataPower is a separate hardware device. When considering the use of IBM WebSphere DataPower in combination with System z, the following key questions should be asked:

- Where are service requests coming from and what is their final destination (the service provider)?

  If both the service requestors and service providers are on System z and WebSphere DataPower is used as the ESB, then each request will have to physically travel through the WebSphere DataPower appliance, meaning travelling twice over the network. This overhead might be outweighed by the benefit of fast transformation processing on WebSphere DataPower.

  If either the service requestor or service provider lives outside System z, this means that a trip over the network has to be made anyway.

- What exact ESB functionality is required?

  WebSphere DataPower provides a significant array of conversion functionality, but cannot be simply compared to an ESB solution such as WebSphere ESB and WebSphere Message Broker. Depending on the requirements, WebSphere DataPower might cover up to 100% of the ESB requirements, especially if these requirements primarily consist of conversion.

A combination of WebSphere DataPower and any of the other ESB solutions discussed earlier can also be a good option. For example, WebSphere

DataPower can be used to convert requests from one format to another, and WebSphere ESB or WebSphere Message Broker can be used purely for advanced mediation and certain specific nodes to access service providers.

One advantage of this solution is that the WebSphere DataPower device® can be used as a security gateway at the same time.

The following scenario is an example of the WebSphere DataPower solution together with WebSphere Message Broker: A requesting application communicates with WebSphere DataPower using SOAP over HTTP, and with the message body encrypted with the WS-Security standard. The DataPower appliance decrypts the body of the message it receives, and the content is then passed to WebSphere Message Broker over a connection secured by HTTPS. WebSphere Message Broker receives the SOAP message and transforms it into a COBOL structure for the final WebSphere MQ application. The responses then flow back in a similar fashion. The initial configuration uses simple HTTP between the WebSphere DataPower appliance and WebSphere Message Broker. The modifications to use HTTPS are performed as a second stage of the configuration.

2.5.5 Using WebSphere MQ as the ESB on System z

WebSphere MQ cannot be considered an ESB solution because it does not provide the rich mediation logic that an ESB requires. WebSphere MQ is a “transport” solution that can be combined with ESB solutions. In fact, many ESB products rely on WebSphere MQ for the inbound and outbound transport of messages. Some custom-built ESBs use WebSphere MQ in combination with a framework. The framework provides routing and some transformation and WebSphere MQ provides the transport.

2.6 Building a federated ESB

Some ISV solutions have their own ESB implementation with the primary purpose of integration between the different functional subcomponents of the same business solution. At the same time, there might be a need to integrate subcomponents or services of these different business solutions. In this case, a layering of ESBs might start to occur, and this is a very likely scenario in any reasonable-size company with at least a few purchased ISV solutions. Typical in this situation is that none of the ESBs “built into” the ISV solutions has sufficient functionality for serving all of the enterprise’s ESB needs.

This “layering” of ESBs is illustrated in Figure 2-27 on page 88. Requests originated in any of the business solutions with a destination in that same business solution in principle run through the ESB of that business solution. Requests originated in any of the business solutions with a destination in another business solution go though the "master" ESB; or, they first go through the business solution ESB and are routed to the master ESB.

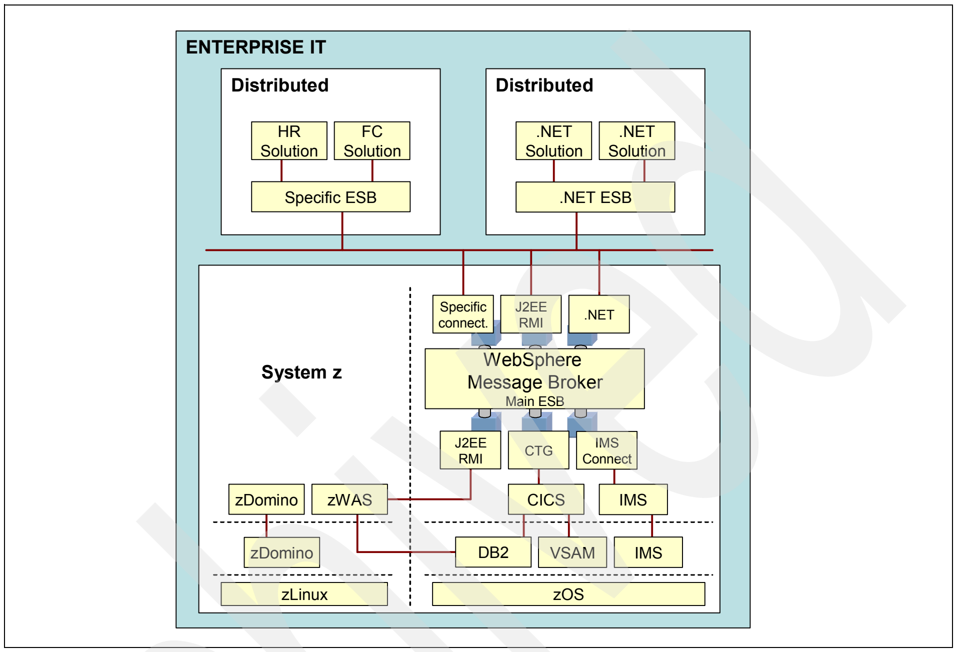


Figure 2-27 Federated ESB example with WebSphere Message Broker

Security

For a federated ESB implementation, it makes sense to consider using WS-Security between the service endpoints rather than between specific pairs of intermediaries. Furthermore, whereas HTTPS with client authentication is implemented at a transport level with no fine level of control for services, WS-Security can be used to provide a high degree of flexibility for addressing security requirements for services, including the authentication of a service request from one ESB Gateway by another.

Standardizing formats

To simplify the design, connectivity between components is handled as though it were another business solution were the ESB is sending a request. A standard data format should be agreed between all components, so every ESB should transform the specific data to the standard before sending a request to another ESB.

2.7 Integrating existing back-end applications

On z/OS, most of the time there are already existing applications with built-in mediation logic. Much of this existing code is dealing with transformation and routing decisions. Business rules are hard coded and this code is hard to maintain and lacks flexibility. The ideal would be to move that type of code gradually over to a dedicated ESB by means of modernization projects, but those projects will be very difficult to justify. One option is to keep ESB logic inside the back-end system and at least try to improve demarcation of the different types of logic in existing CICS and IMS applications, but better would be to start connecting back-end systems to an ESB and expose back-end functions as a service.

IMS

The recommended architecture is to connect the IMS system to the ESB layer, and choose an integration mechanism and protocol between the ESB and IMS. It is also very important to think about the granularity of the IMS components to be accessed. In many cases, refactoring of IMS applications will be necessary to achieve optimal service granularity and resolve demarcation issues. This topic will be addressed extensively in a future Redbooks publication on z/OS application modernization.

IMS TM provides an interface layer called OTMA (Open Transaction Manager Access). Normally, systems do not interact directly with the OTMA interface, but OTMA-capable connectors are used. Currently there are two connectors to access IMS using this channel: IMS Connect and the MQ Bridge for IMS.

Figure 2-28 on page 90 shows a simplified view of the integration options for IMS transactions and IMS data.

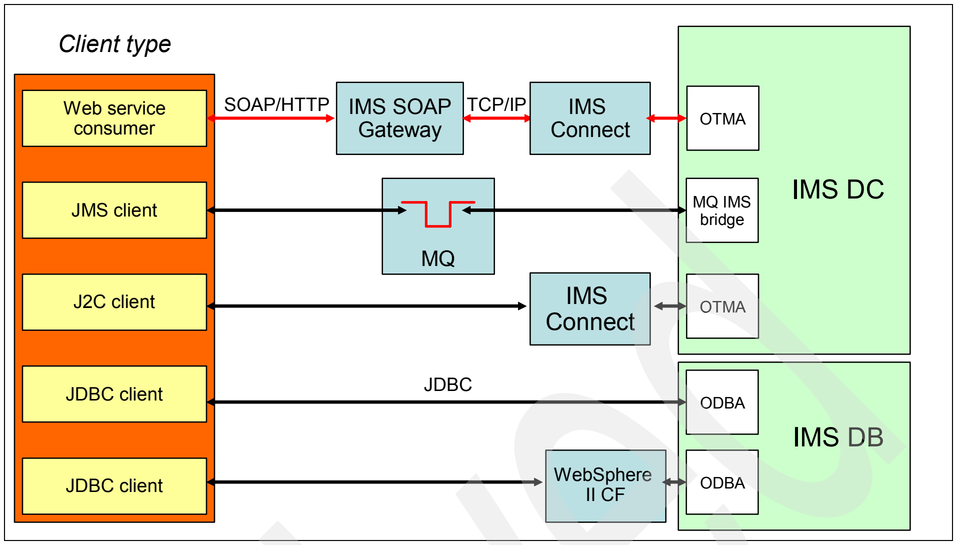


Figure 2-28 IMS integration options

IMS Connect through a J2C connector

One option to access IMS Connect is through a J2C connector from a J2EE application. This option requires you to install the J2C resource adapter for IMS in the J2EE application server or the ESB. When using WebSphere ESB as the ESB, the ESB flow will provide the protocol transformation between the incoming SOAP/HTTP or SOAP/JMS request and the J2C protocol, and it also will map the message between the SCA interface, representing the connector, and the exported SOAP/HTTP or SOAP/JMS interface.

At the time of writing WebSphere Message Broker does not support J2C connectors.

IMS Connect through IMS SOAP Gateway

IMS Connect can also be accessed directly from a Web Service requestor through the IMS SOAP Gateway. In this case the service requestor sends a SOAP/HTTP request directly to the IMS SOAP Gateway and the IMS SOAP Gateway takes care of any transformation issues between the incoming format (typically, a SOAP message with XML content) and IMS Connect.

Figure 2-29 on page 91 shows the architecture of the IMS SOAP Gateway.

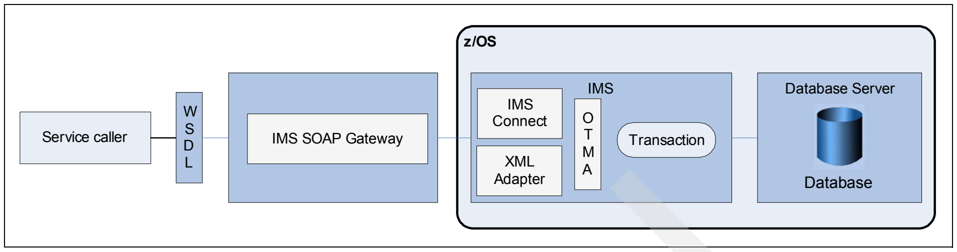


Figure 2-29 IMS SOAP Gateway

OTMA using a custom interface

It is also possible to build a customized connector that uses the OTMA interface directly to access the IMS system, but the recommended solution will be to use one of the standard connectors unless you have requirements that lead to building a new one, like performance, security, and so on.

MQ Bridge for IMS

The MQ Bridge for IMS provides access to IMS programs using queues to send and receive requests in an asynchronous manner. Any WebSphere MQ client can make use of this mechanism to send a request to an IMS program and receive a response back. When used from an ESB, it is necessary that the ESB can work with WebSphere MQ queues. This is the case for most ESB solutions, including WebSphere ESB and WebSphere Message Broker. The Web Service interface can be implemented in the ESB itself. This architecture is shown in Figure 2-30.

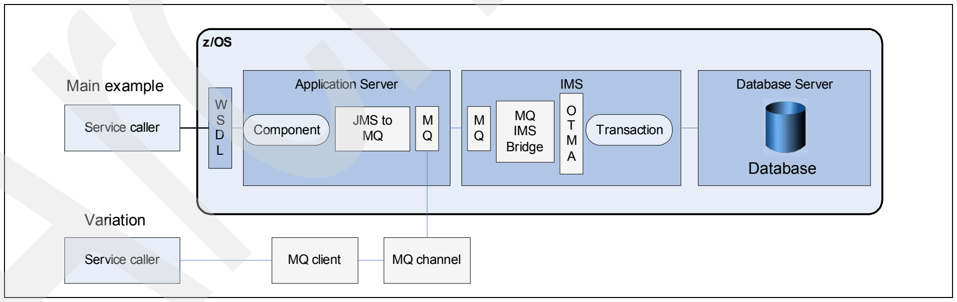


Figure 2-30 MQ Bridge for IMS

CICS

The recommended architecture is to connect the CICS system to the ESB layer, and choose an integration mechanism and protocol between the ESB and CICS. It is also very important to think about the granularity of the IMS components to be accessed. In many cases, refactoring of CICD applications will be necessary to achieve optimal service granularity and resolve demarcation issues. This topic will be addressed extensively in a future Redbooks publication on z/OS application modernization.

CICS provides several interfaces to the outside world, as shown in Figure 2-31.

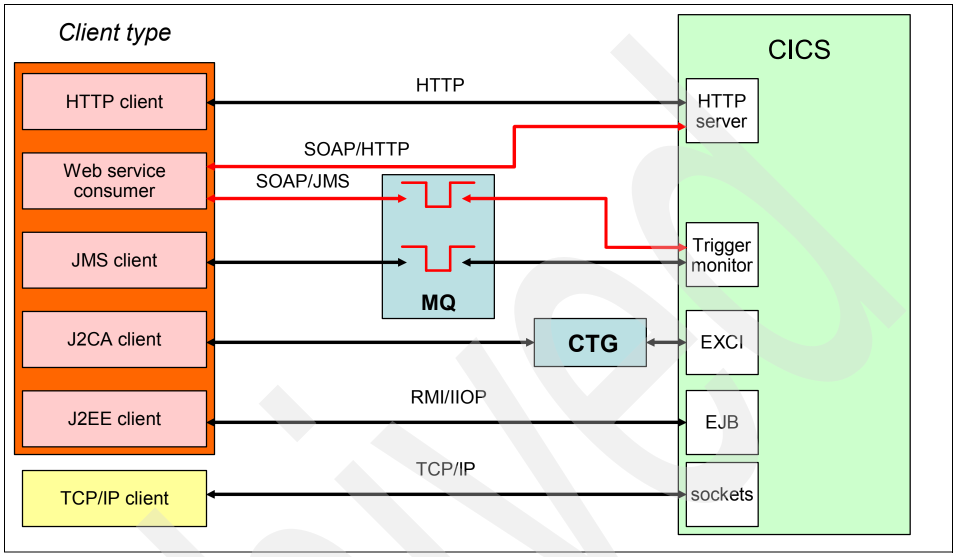
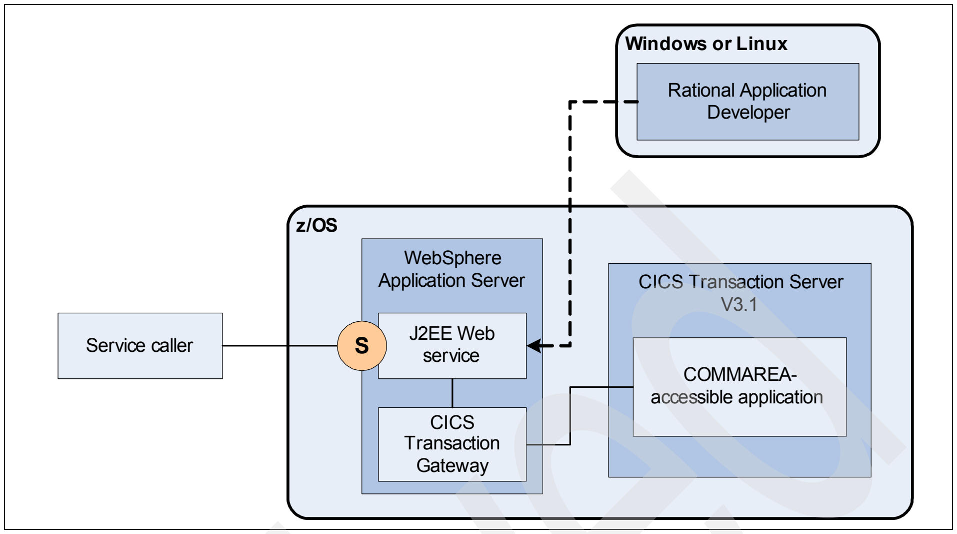


Figure 2-31 CICS integration options

CICS Transaction Gateway through a J2C connector

Any CICS business logic program can be accessed from outside using a native CICS interface. When accessed locally, this is *External CICS Interface (EXCI)*, when accessed remotely, it is either *External Call Interface (ECI)* or *External Presentation Interface (EPI)*, depending on whether the program has an associated BMS map or not. These interfaces are driven by the *CICS Transaction Gateway (CICS TG)*. Client programs on their turn work with the available CICS TG APIs.

Figure 2-32 on page 93 shows the CICS TG architecture when used from WebSphere Application Server. In this diagram the Web Service interface is implemented in WebSphere Application Server, but this could also be WebSphere ESB.


Windows or Linux
Rational Application Developer
z/OS
WebSphere Application Server
J2EE Web service
S
CICS Transaction Gateway
CICS Transaction Server V3.1
COMMAREA-accessible application
Service caller
Figure 2-32 CICS Transaction Gateway

CICS TG is a proven integration solution between the Java world and CICS. Recent versions provide extensive security functions and 2-Phase Commit support.

CTG APIs can be used from stand-alone Java programs and full J2EE application servers. When used in a J2EE application server, the CICS TG J2C resource adapter is installed into the application server.

From an ESB perspective, WebSphere ESB supports the CICS TG J2C connector and WebSphere Message Broker does not.

CICS Web Services

CICS provides support for direct inbound and outbound Web Services calls. Thus, service consumers can send their SOAP request directly to CICS, either using an HTTP or JMS transport. Of course, an ESB can be used to route those SOAP requests into CICS. Conversely, CICS programs can be a service consumer themselves and perform an outbound Web Service call to an external Web Service. Most ESB solutions support Web Services standards, both inbound and outbound.

Figure 2-33 on page 94 gives an overview of CICS Web Services inbound. Note that both an HTTP channel and a JMS channel are present.

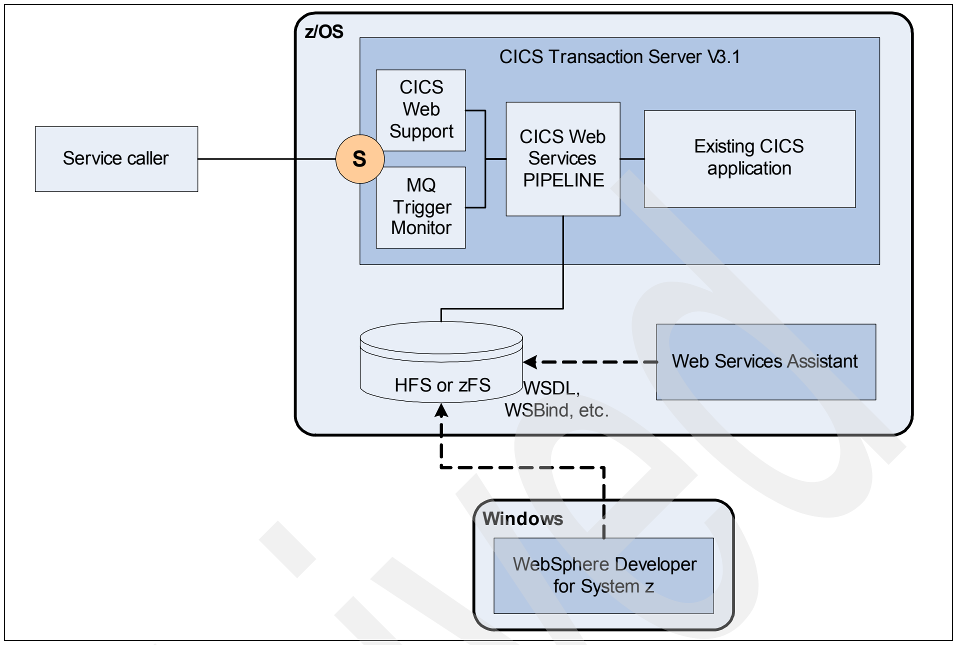

z/OS
CICS Transaction Server V3.1
CICS Web Support
MQ Trigger Monitor
S
CICS Web Services PIPELINE
Existing CICS application
Service caller
HFS or zFS
Web Services Assistant
WSDL, WSBind, etc.
Windows
WebSphere Developer for System z

Figure 2-33 CICS Web Services inbound

MQ Bridge for CICS

The MQ Bridge for CICS provides access to CICS programs using queues to send and receive requests in an asynchronous manner. Any WebSphere MQ client can make use of this mechanism to send a request to a CICS program and receive a response back. When used from an ESB, it is necessary that the ESB can work with WebSphere MQ queues. This is the case for most ESB solutions, including WebSphere ESB and WebSphere Message Broker. The Web Service interface can be implemented in the ESB itself. This architecture is shown in Figure 2-34.

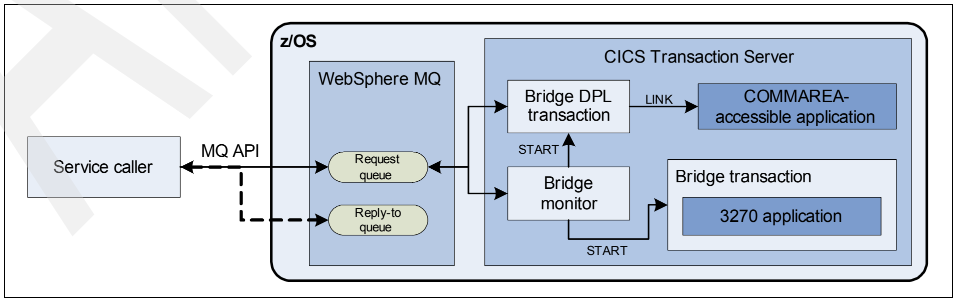


Figure 2-34 MQ Bridge for CICS

J2EE RMI-IIOP connectivity

CICS EJB™ programs can be accessed through an RMI-IIOP call from a J2EE client. This J2EE client can be wrapped into a Web Service implemented in a J2EE application server and thus provide the service interface to a back-end CICS EJB.

2.8 WebSphere Service Registry and Repository

IBM WebSphere Service Registry and Repository (WSRR) is a tool that enables better management and governance of services through its registry and repository capabilities and its integration with the SOA architecture. It enables you to store, access, and manage information about services and service interaction endpoint descriptions (referred to as service metadata) in an SOA. This information is used to select, invoke, govern, and reuse services as part of a successful SOA.

This service information includes traditional Web services that implement Web Services Description Language (WSDL) interfaces with SOAP/HTTP bindings, as well as a broad range of SOA services that can be described using WSDL, XML Schema Definition (XSD), and policy declarations, but might use a range of protocols and be implemented according to a variety of programming models, like CICS, COBOL, and IMS.

At runtime, WSRR service queries through the API allow infrastructures such as the ESB to make dynamic endpoint selection decisions based on metadata in the registry. It has the ability to query and find the service metadata that is used through tooling for reuse of interfaces and schemes as well as static binding of service endpoints.

WebSphere Message Broker, WebSphere ESB, WebSphere DataPower, and CICS all provide support for accessing WSRR. Figure 2-35 on page 96 shows the concept of dynamically looking up service information in WSRR from a mediation running in an ESB.

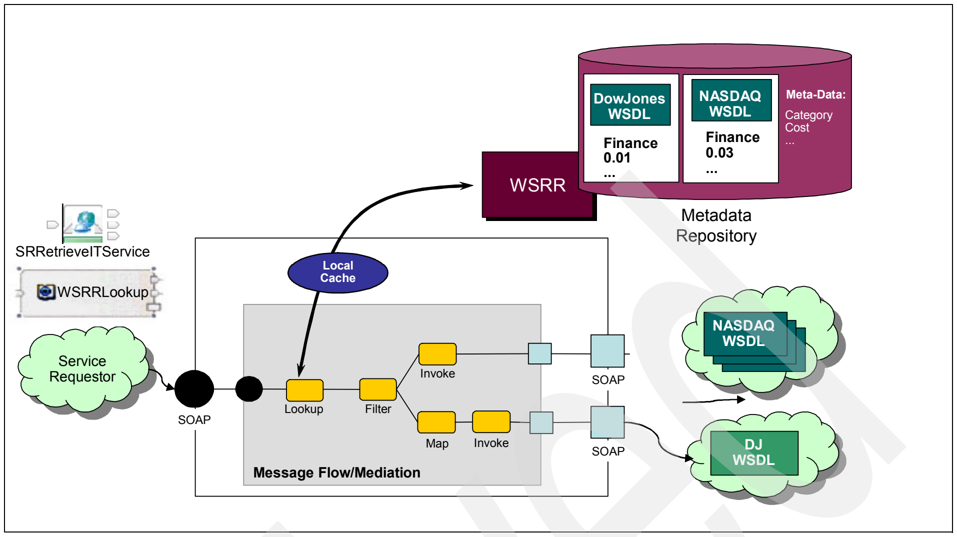


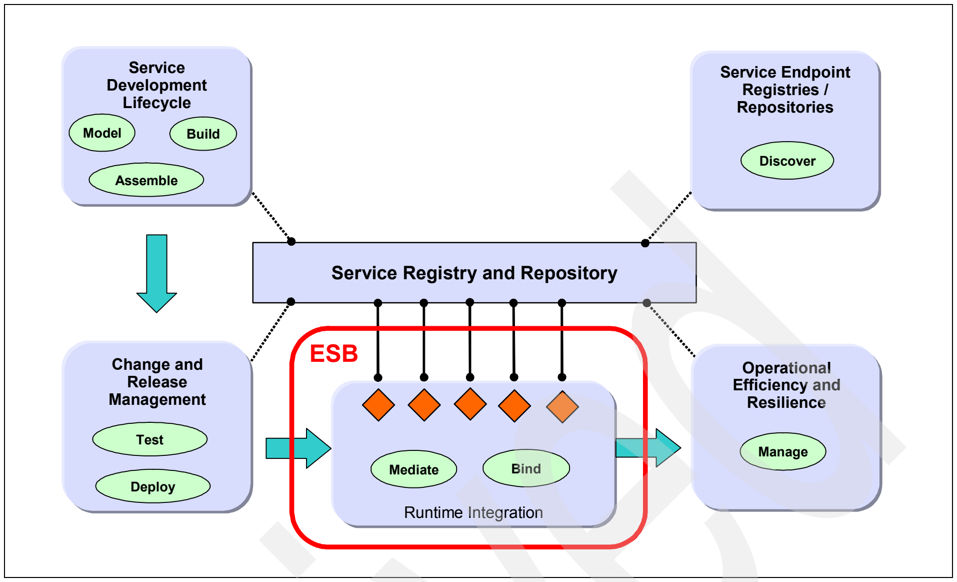
Figure 2-35 Dynamic lookup of service information in WSRR

WebSphere Service Registry and Repository is used to store information about services in systems, or in other organizations' systems, that we already use, plan to use, or that we want to be aware of. For instance, an application can check with WSRR just before it invokes a service to locate the most appropriate service that satisfies its functional and performance needs. This capability helps make an SOA deployment more dynamic and more adaptable to changing business conditions.

WebSphere Service Registry and Repository includes:

- A service registry that contains information about services, such as the service interfaces, its operations, and parameters
- A metadata repository that has the robust framework and extensibility to suit the diverse nature of service usage

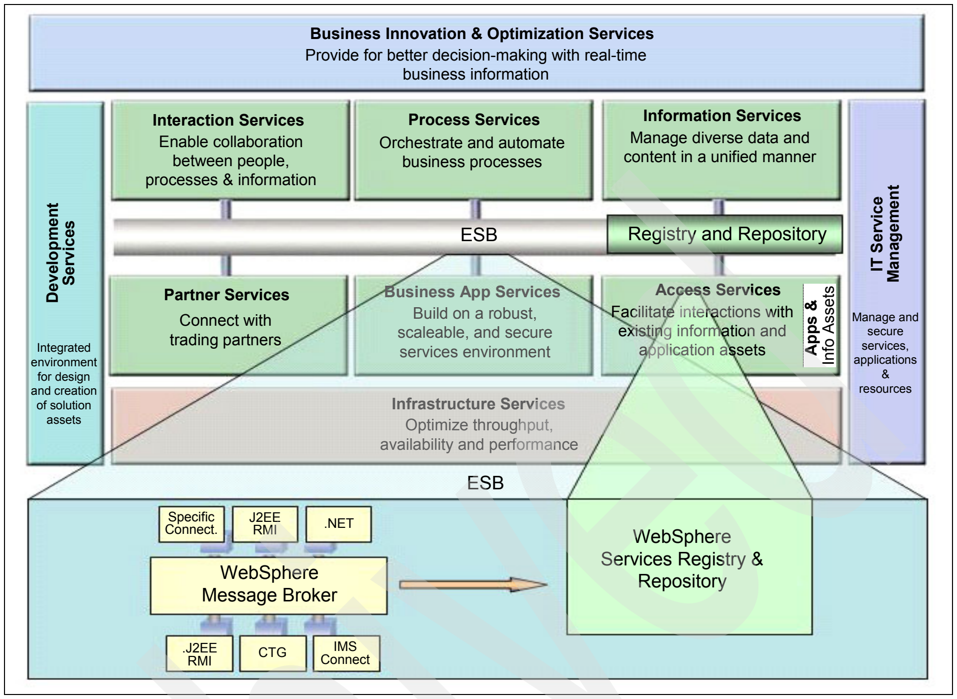
To illustrate how to use WebSphere Service Registry and Repository across the SOA life cycle phases, we now take a look at a day in the life of a business service — from its invention in the model phase of the SOA life cycle, to its translation into an executable composite application in the assemble phase, to its transition into the SOA run-time environment in the deploy and run phase, to it being observed and controlled in the manage phase of that life cycle (see Figure Figure 2-36 on page 97).



Service Development Lifecycle
Model
Build
Assemble
Service Endpoint Registries / Repositories
Discover
Service Registry and Repository
Change and Release Management
Test
Deploy
ESB
Mediate
Bind
Runtime Integration
Operational Efficiency and Resilience
Manage

Figure 2-36   WebSphere Service Registry and Repository functionality

An ESB uses a service registry to obtain metadata about the services it needs to invoke. This metadata can be specific service endpoints, supported port types, services that meet a certain SLA requirement, or other metadata. In Figure 2-37 on page 98, we can see that when a mediation is invoked on the ESB (Message Broker, in this example), the ESB processes the incoming message using specific nodes to access the necessary metadata from the service registry (WebSphere Service Registry and Repository). The ESB then uses the retrieved metadata to set the correct endpoints to invoke, determine alternate execution paths, enforce policy, or perform other actions.



Business Innovation & Optimization Services
Provide for better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enable collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manage diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
ESB
Registry and Repository
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitate interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimize throughput, availability and performance
ESB
Specific Connect.
J2EE RMI
.NET
WebSphere Message Broker
J2EE RMI
CTG
IMS Connect
WebSphere Services Registry & Repository

Figure 2-37 Identifying WebSphere Service Registry and Repository into the ESB

Understanding registry contents

The most elemental building blocks for the WebSphere Service Registry and Repository content model are service metadata artifact documents (physical documents), such as XSD or WSDL files. These service metadata documents are stored and managed in WebSphere Service Registry and Repository. Any service metadata artifact type can be stored in WebSphere Service Registry and Repository; however, WebSphere Service Registry and Repository offers advanced functions for the following well known SOA metadata types: WSDL, XSD, WS-Policy, and SCDL. The contents of these types of service metadata documents comprise the logical model of the WebSphere Service Registry and Repository. The logical model has entities such as portType, port, and messages related to WSDL files, and complexType or simpleType related to XSD documents. Elements of the logical model have properties and relationships reflecting a subset of their characteristics as defined in the underlying physical document. For example, a WSDLService element has a namespace property and a relationship to the ports it contains.

There is one other kind of entity in the WebSphere Service Registry and Repository, loosely referred to as a “Concept,” which is a generic object that can be used to represent anything that does not have a physical document in the

WebSphere Service Registry and Repository. Concepts can be used to represent a reference to content in some other metadata repository, such as a portlet in a portlet catalog, an asset in an asset repository, service implementation artifacts kept in a source code library, or information about SOA infrastructure topologies in a configuration management database. They can also be used to group physical artifacts together for ease of retrieval, for example.

In addition to content directly related to service metadata documents, WebSphere Service Registry and Repository supports a number of user-defined metadata types that are used to decorate the service metadata to explain their semantics; these are referred to as “Service Description Metadata.” WebSphere Service Registry and Repository supports three types of service semantics metadata types: Properties, Relationships, and Classifiers. Users can associate a property businessValue with a physical model entity representing a WSDL file, or define a new relationship makesUseOf between an entity in the logical model representing a portType and an entity in the physical model representing an XML document; or create a classifier importantThings and associate it with a port entity in the logical model and with an entity in the physical model representing a policy document. This enables semantic queries to target individual elements of the service metadata, and meaningful dependency analysis to take place prior to making changes.

The main user interface for WebSphere Service Registry and Repository is a Web application deployed with the WebSphere Service Registry and Repository runtime. This interface supports a set of anticipated user roles and provides the following capabilities:

- Look-up, browse, and retrieve
- Publish and annotate
- Governance activities, including import/export and impact analysis

2.9 Conclusions

Clearly, an ESB is a mandatory logical layer in an SOA. An ESB is the way to go to enhance flexibility in integration, increase reusability, and thus reduce complexity. We have seen that there are a variety ESB solutions possible on System z and an ESB does not necessarily have to run on one platform. In a complex IT landscape, it is very likely we will see ESB networks with hubs on multiple platforms. Because an ESB will grow into an extremely mission-critical IT component, platform choice is a serious matter and System z should be a serious candidate.

3

Business Process Services

In this chapter, we describe the functions provided by Business Process Services and its role within the Business Process Management (BPM) life cycle. We present some background about Business Process Management. We also describe some architectural viewpoints, suggested infrastructure patterns, and possible solution scenarios for the Business Process Services implementation.

3.1 Business Process Services

Business Process Services is part of one of the five major steps in the *Business Process Management (BPM)* life cycle in SOA. As shown in Figure 3-1, the BPM life cycle consists of the steps *Model*, *Assemble*, *Deploy*, *Manage*, and *Governance*. Business Process Services fit into the Deploy step. Business Process Services integrate people, integrate processes, and manage and integrate information.

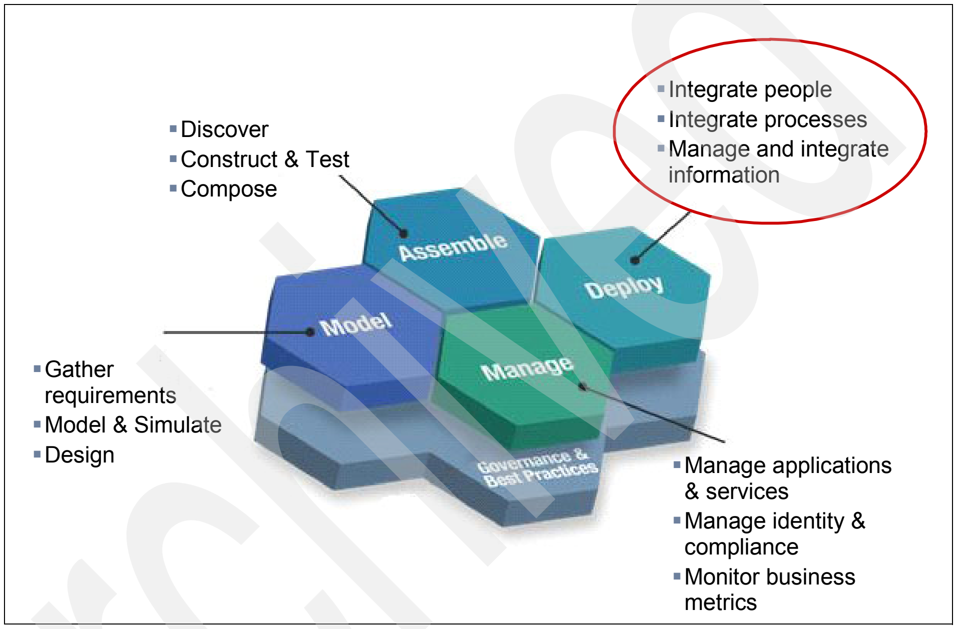


Figure 3-1 Business Process Services in BPM life cycle

3.1.1 BPM introduction

Before we discuss the Business Process Services definitions, scope, and functions, we start with a high level view of Business Process Management (BPM). Business Process Management is a discipline combining software capabilities and business expertise to accelerate process improvement and facilitate business innovation. The Business Process Management life cycle consists of Design, Modeling, Execution, Monitoring and Optimization. BPM enabled by SOA is a discipline enhanced by a flexible IT architecture to accelerate the creation and reuse of business services to facilitate business innovation. Bringing together the most advanced SOA-based software capabilities along with broad expertise will provide a higher value BPM solution.

Figure 3-2 shows a high level view of BPM.

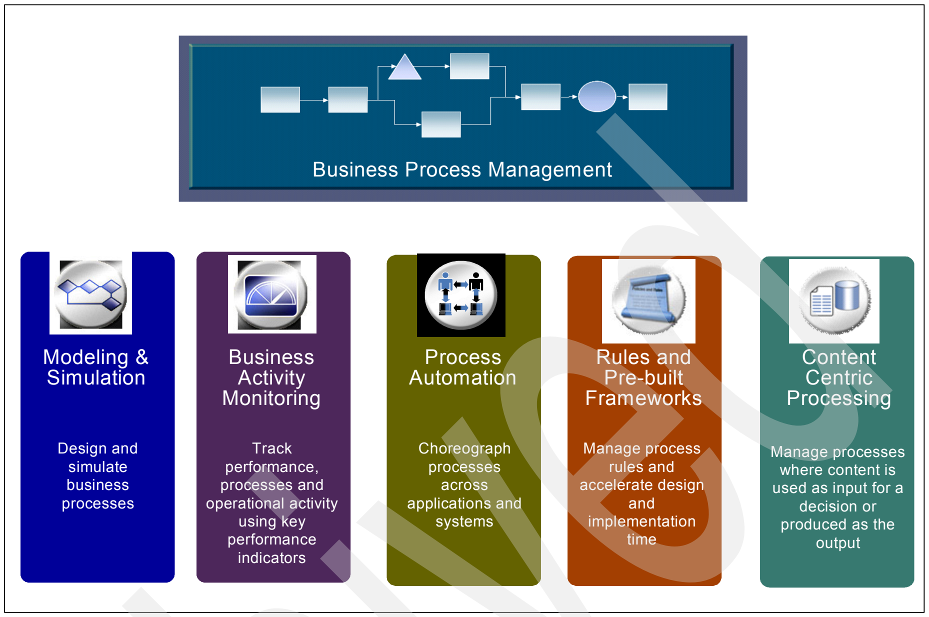

Business Process Management
Modeling & Simulation
Design and simulate business processes
Business Activity Monitoring
Track performance, processes and operational activity using key performance indicators
Process Automation
Choreograph processes across applications and systems
Rules and Pre-built Frameworks
Manage process rules and accelerate design and implementation time
Content Centric Processing
Manage processes where content is used as input for a decision or produced as the output

Figure 3-2 Business Process Management: High level view

BPM drives business and IT alignment around the following functional and transformational business objectives:

- Collaborate to predict and optimize process outcomes through modeling and simulation.
- Rapidly customize processes with business users using policies instead of code.
- Sense and respond to business events in real time for automated response or human decision support.
- Rapidly deploy new solutions from reusable building blocks that can be changed on-the-fly.

BPM with SOA provides process flexibility by improving the way you design, manage, and optimize the business processes and reuse existing assets. BPM enabled by SOA is based on creating agile and dynamic processes today that serve as the foundation for greater innovation in the future. It provides the key alignment between business architecture and IT infrastructure and keeps this alignment flexible and continuous. BPM offers a service-oriented approach to application development, based on orchestration of services. You can take existing software assets and quickly define how those assets are used in a new integration application.

Figure 3-3 displays the relationship between BPM and SOA.

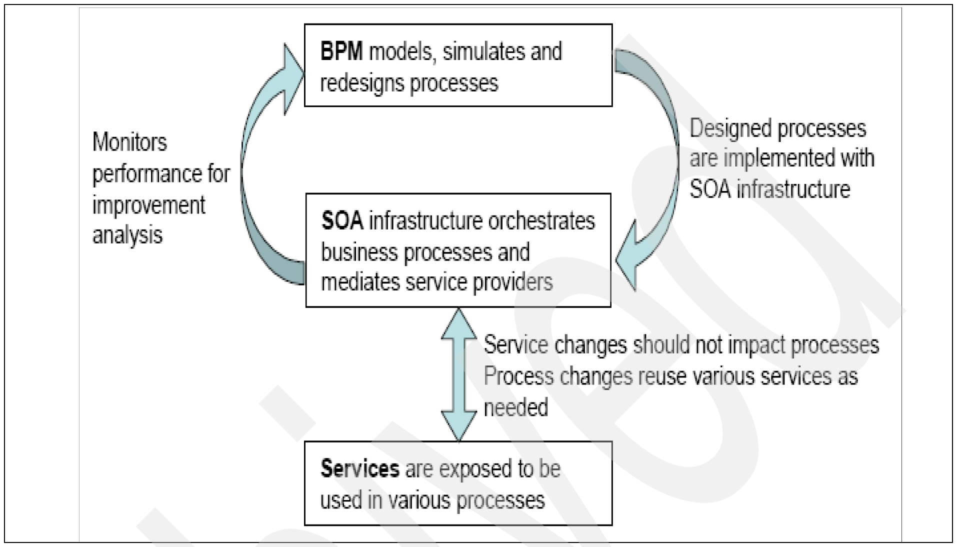

BPM models, simulates and redesigns processes
Monitors performance for improvement analysis
Designed processes are implemented with SOA infrastructure
SOA infrastructure orchestrates business processes and mediates service providers
Service changes should not impact processes Process changes reuse various services as needed
Services are exposed to be used in various processes

Figure 3-3 Relationship between BPM and SOA

The benefits of BPM

BPM as a software technology attacks these challenges head-on. It does not replace your existing IT investments but coordinates their actions to make end-to-end processes more efficient, more flexible and agile, and more standardized and compliant. BPM automates, integrates, and optimizes business processes. It does that by defining process models or templates that specify the flow of activities, both human and automated, and then managing that flow and continuously monitoring its performance.

- It is *efficient* because it automates manual tasks and handoffs, and makes sure the most important tasks are done first and on time. In applications as diverse as loan origination, claims processing, telco service provisioning, customer service, and billing dispute resolution, BPM can cut cycle times by over 70% and commonly enables dramatic increases in processing volume with no addition to headcount.
- It is *agile* because executable process models are not built with complex code but composed graphically like a flowchart, so they can be built quickly and easily changed. A key enabler of BPM has been the emergence of service-oriented architecture, which allows the process to integrate discrete activities of external applications without writing code. This accelerates

time-to-market with new offerings and allows exiting processes to respond instantly to new competitive demands.

- It is *compliant* because process logic is based on rules reflecting policies and best practices. Process components based on those rules can be shared and reused across the organization, so the rules and standards are always followed in every office of your enterprise. Moreover, because BPM records every step performed, processing is auditable from start to finish.
- It makes processes *visible* end-to-end by aggregating data from disparate business systems along with human workflow statistics, and displaying key performance indicators in management and administrative dashboards. Performance management allows process bottlenecks to be relieved in real time, while maintaining a secure audit trail to ensure provable compliance.

The automation, integration, and visibility BPM brings to cross-functional processes translate into tangible return on investment. BPM makes processes run faster and employees more productive. Besides automating manual procedures, it streamlines handoffs and optimizes tasks by priority. It also provides deadlines, notifications, and user-defined escalation actions that put the focus on customer value rather than simply first-in first-out.

BPM also brings standardization and control. You can enforce business rules, and prove to an auditor that you did. You can replicate best practices globally across all offices in the enterprise. You can track service level agreements (SLAs) and compliance with regulations, even when the process spans departments or possibly outsourced functions. Exceptions are handled explicitly within the process model, and all actions are logged automatically for performance management and auditability.

Moreover, BPM lowers the cost of developing and maintaining business solutions. Because executable processes can be designed and maintained with little programming and process model components can be reused easily in new variants, BPM reduces the demand on critical IT resources as well as total cost of operations.

Beyond those quantifiable benefits, BPM adds strategic value as well, for example:

- Agility: The ability to bring new products and services to market quickly and to respond rapidly to changing demands. While enterprise applications are hard to change, the process model that coordinates their actions is easy to change.
- Business integration: Through its standards-based integration middleware, BPM allows you to extend the scope of process automation and management across the IT barriers that historically have separated departments, front and back office, and your customers and suppliers. This can be done despite the

diversity of system platforms, API languages, and data models they represent.

- Global visibility: BPM makes process performance visible at the process level, tracking process data and aggregating it in tables of key performance indicators and graphical dashboards at various levels for process owners, system administrators, and business executives.

3.1.2 Definitions

A *business process* is a collection of business activities fulfilling customer's business needs. *Business activities* have clearly defined boundaries, and they take input and produce output that is of value to the customers.

Business processes can be cross-functional and can span across business units or organizations. Each process has a clear input and output defined. Transformation can take place within the process and add value to the recipients. Activities within the process are ordered (*orchestrated*) with respect to their timing and functional requirements. Processes should be independent of specific information resources and specific task automation applications. The integration technology must loosely couple the applications and resources that make up the process, otherwise the logic of a process will get hard-coded into a particular technology platform.

Business process types

There are several ways to differentiate types of business processed. A common way is to make a distinction between *long-running,* with eventual human tasks, and *short-running,* without human tasks.

Long-running processes

A long-running business process is interruptible, and each step of the process can run in its own physical transaction. Long-running business processes can wait for external stimuli. Examples of external stimuli are events that are sent by another business process in a business-to-business interaction, responses to asynchronous invocations, or the completion of a human task. A long-running process has the following characteristics:

- It runs as several transactions.
- It consists of synchronous and asynchronous services.
- It stores each intermediate process state, which makes the process forward-recoverable.

Some people refer to this type of process as a *macroflow*.

Short-running processes

A short-running process runs in one physical thread from start to finish without interruption. Short-running processes are sometimes referred to as non-interruptible business processes or *microflows*. These processes can have different transactional capabilities. A process can run within a global transaction or as part of an activity session. This type of process has the following characteristics:

- It runs in one transaction.
- It normally runs for a short time.
- It does not store run-time values in the database.
- It consists of only synchronous services and non-interruptible subprocesses, which means that a short-running process or micro flow cannot contain:
  - Human tasks
  - Wait activities
  - Multiple initiating receive activities
  - Non-initiating receive activities

A short-running process or microflow should not invoke the following services or activities:

- Long-running services
- Activities bound to asynchronous protocols

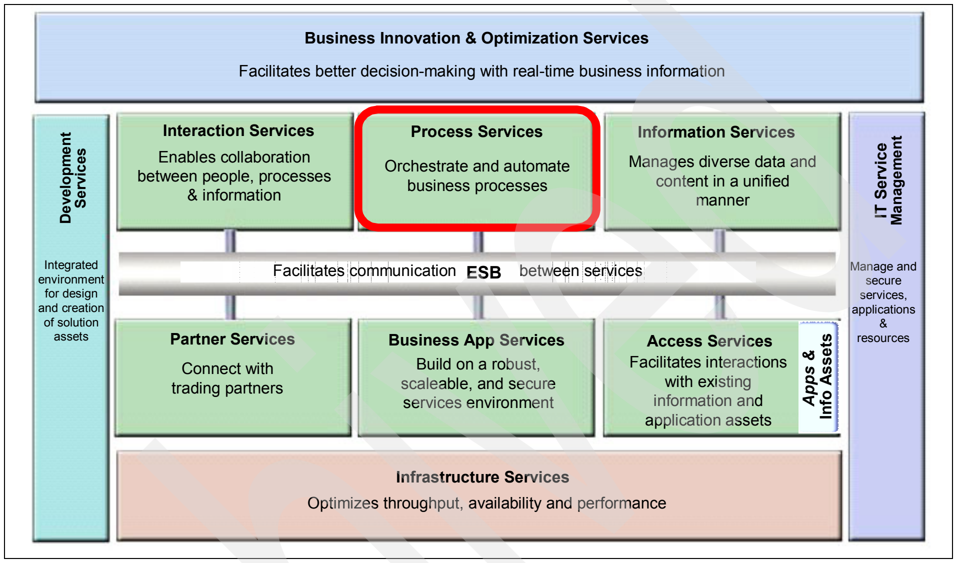
Human tasks

A *human task* is a component that involves a person interacting with a service or another person. The interaction can be initiated either by a person or by an automated service. A service that is initiated by a person can be either an automated implementation or a service that is provided by another person. A human task that is invoked by an automated service can be replaced easily by an automated implementation, and vice versa.

3.1.3 Scope

Business Process Services include various forms of compositional logic, the most notable of which are business process flows and business state machines (finite-state machines for business composition). We consider both kinds of composition mechanisms, plus other forms such as business rules and decision tree processing, as well as more ad-hoc forms of composition, to be equally valid approaches to choreographing service composition.

Business Process Services is one of the major components within the Business Process Management life cycle. *Business Process Management Solution (BPMS)* supports the end-to-end process life cycle from modeling to performance management and optimization based on BPEL and SOA. In addition, Process Services is one of the components in the SOA Reference Architecture shown in Figure 3-4.



Business Innovation & Optimization Services
Facilitates better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enables collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manages diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
Facilitates communication ESB between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitates interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimizes throughput, availability and performance

Figure 3-4 Business Process Services in SOA

3.1.4 Functions

Business Process Services provides control and management services that allow integration and automation of the business processes. The integration technology must loosely couple the applications and resources that make up the process; otherwise, the logic of a process will get hard-coded into a particular technology platform, which can be expensive to change and defeats the entire purpose of BPM. This is where standards-based service-oriented architecture comes in. SOA provides the technical ability to create process independence. SOA standards, such as Web Services, make information resources and task automation applications available, yet loosely integrated, for process designers to use and reuse at will.

Services are exposed as Web services, which will be used by the various business processes. SOA enables process infrastructure driven choreography of services. Services changes should not impact processes. Process changes reuse various services as required. Business Process Services combines various forms of services, compositional logic that spans people, business applications, different systems and platforms, and links them together into single orchestrated workflow. Business rules and human interaction are also part of the solution.

Business process orchestration contains activities that include, among others, the following:

- Business process modeling
- Process choreography
- Business rules implementation
- Event management

A business process can consume service component architecture (SCA) services or it can be consumed by other SCA services.

3.2 Key architectural decisions with respect to business process services

In this section, we present a checklist to lead you through the architectural decisions that must be made with respect to business process services:

- Decide if BPM and a business process services solution will meet the enterprise directions and business requirements, including the short term business requirements and the long term business strategy. Things to be considered are:
  - Is there an enterprise vision of BPM and is the value clearly understood?
  - Is BPM already in place in the enterprise and how is it supported?
  - Is there a path for future growth in BPM?
  - If there is an existing BPM solution, how can it be enhanced and expanded into other areas (business units and business partners)?
  - If there is no BPM solution, will buying a packaged BPM solution meet the business objectives and requirements?
  - Are there any alternatives?
- Decide on the application patterns to meet the business process services requirements. It is likely multiple patterns will need to be supported. Refer to “Business process services location patterns” on page 112 for more details. Things to be considered are:
  - Will there be *serial* workflows or *parallel* workflows?
  - Will there be any *Straight-Through Processing (STP)* processes?

  - Will there be people intervention in the processes?
  - Will there be any communication with other services partners via the Internet or intranet?
  - Will there be any integration with other services?
- Decide on the desired location pattern for the business process services implementation. Refer to “Business process services location patterns” on page 112 for more details. Things to be considered are:
  - Are the business process services close to the enterprise business data?
  - Where does the enterprise business data reside?
  - Will there be any integration with other business process services?
  - Will the business process services solution need to be close to the services it depends on?
  - Where do other services reside?
    This can be on another platform, another datacenter or even at a business partner location.
- Decide on the platform where the business process services will be deployed. Things to be considered are:
  - What kind of platform infrastructure currently exists?
  - Will there be sufficient capacity to run the new business process services solution?
  - Are hardware upgrades required for the business process services solution and what will be the cost associated with this?
  - Are software upgrades required to support the business process services solution and what will be the cost associated with this?
  - How does the target platform meet the business process services functional requirements?
  - How does the target platform meet the business process services non-functional requirements, such as performance, high availability, scalability, and reliability?
  - What level of Quality of Services is required by the business process services and how does the target platform meet the QoS requirements such as security, workload management, backup and restore capabilities?
  - Are the required skills available for supporting the target platform for the business process services?
- Decide on the business process services development tools and modeling tools. We do not discuss this aspect further in this chapter, but things to be considered are:

- What development tools and modeling tools already exist in the organization?
- Will those tools meet the business process services development requirements?
- What additional development tools and modeling tools are required for the new business process services?
- Are the required skills available for the business process services development?

► Decide on the product to implement the business process services solution. Things to be considered are:

- Are current assets (business process services) being reused?
- Buy a business process services product or build one (a framework)?
- What are the costs associated, including one-time costs and on-going maintenance costs?
- Are the required skills available?

► Decide on the business process services deployment topologies with respect to the platform chosen. Things to be considered are:

- How many LPARs are needed if the solution will reside on z/OS?
- How many z/VM LPARs and Linux servers are needed if the solution will reside on Linux on System z?
- How many servers are needed if the solution will reside on distributed systems?
- How much disk space is needed to deploy the solution?
- Is clustering of business process services servers required?
- Are there any networking requirements?
- Are there any disaster recovery requirements?
- Are there separate deployment environments for DEV/TEST/UAT/Production?

► Decide on the business process services integration solution with respect to other Web services, applications, and business processes. Things to be considered are:

- Are the business process services close to the services they depend on?
- Are the other business applications Web Services enabled?
- What is the current communication mechanism (interface) of the existing business applications?
- Will the existing business application need to be modernized?

  - How will the current business applications be modernized?
  - Are the required skills available for application modernization?

3.2.1 Some key architectural principles

Following are several architectural principles to bear in mind when designing the business process services solution:

- Applications should be delivered as sets of services rather than monolithically.
- Services must leverage and integrate current applications in order to create a seamless business process flow.
- Service interfaces between the service requesters and service providers should be loosely coupled.
- *Separation of concerns* must be promoted. This means separating application functions from infrastructure and encouraging reuse.
- Services should be composable such that higher level services can be constructed from the lower level services.
- Service interfaces must be platform independent.
- Services should have a high level of reusability, and thus an optimal *granularity*.
- Business processes must be close to the enterprise business data.
- The business process solution should be close to the services it depends on.
- Aim to reduce total cost of ownership and provide easier management, for example, through virtualization.

3.3 Business process services location patterns

This section describes five location patterns that can be used to deploy the Business Service Choreography server. The business processes deployed can be short-running *microflows* or long-running processes. The patterns are:

- Business process services on z/OS
- Business process services on Linux on system z
- Business process services on distributed systems
- Business process services on z/OS and distributed systems
- Business process services on Linux on System z and distributed systems

3.3.1 Business process services on z/OS

This pattern describes the business process services residing on z/OS. All application patterns described in Appendix A, “Business Process services application patterns” on page 481 can apply to this location pattern. Both the microflows and long-running processes are running on the same location, which is on z/OS. Figure 3-5 shows the business process residing on z/OS, but it also consumes several other services on the distributed systems.

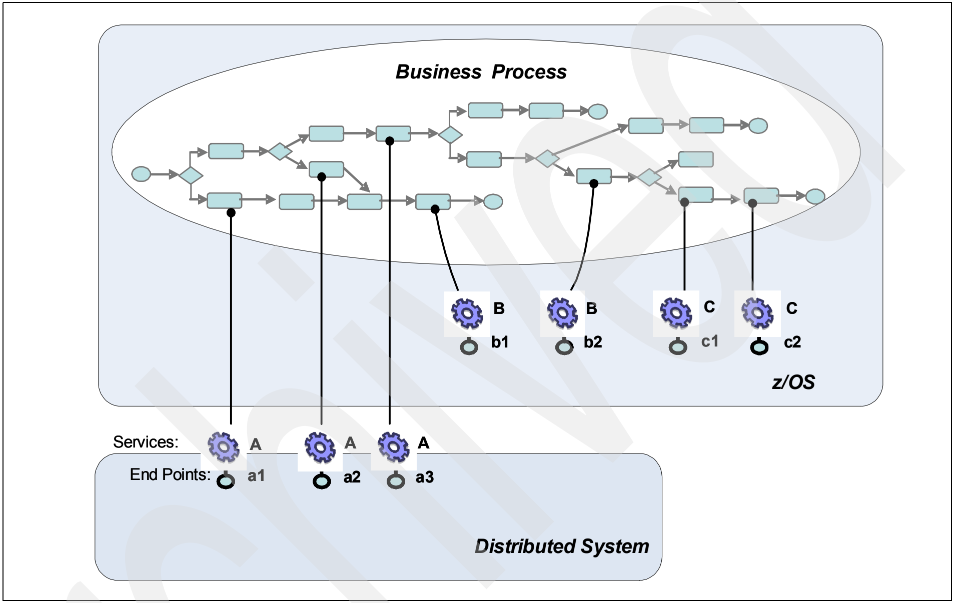


Figure 3-5 Business process services on z/OS

Benefits and drawbacks

This pattern leverages all the benefits z/OS and the IBM system z platform provide. It also benefits from the business processes being close to the enterprise application services they depend on. The cost of running incremental workload on the mainframe goes down as the total workload grows. It is more logical to run the Business Process Engine on the same platform location on which most of the business application services reside. In a typical IBM mainframe environment, many of those business application services will be running in CICS or IMS or both. For long-running processes, a highly available Business Process Engine database is critical in order to ensure the integrity of the process states. DB2 on z/OS in combination with Parallel Sysplex on z/OS provides the best in industry database high availability technology and can be used to run the Business Process Engine.

3.3.2 Business process services on Linux on System z

This pattern describes the business process services residing on Linux on System z. All application patterns described in Appendix A, “Business Process services application patterns” on page 481 can apply to this location pattern. Both the microflows and long-running processes are running on the same location, which is Linux on System z in this case. Figure 3-6 on page 114 shows the business process residing on Linux on System z, but it also consumes services on the z/OS system, the distributed systems, and another Linux on System z server. The business application services on z/OS could be CICS and IMS applications.

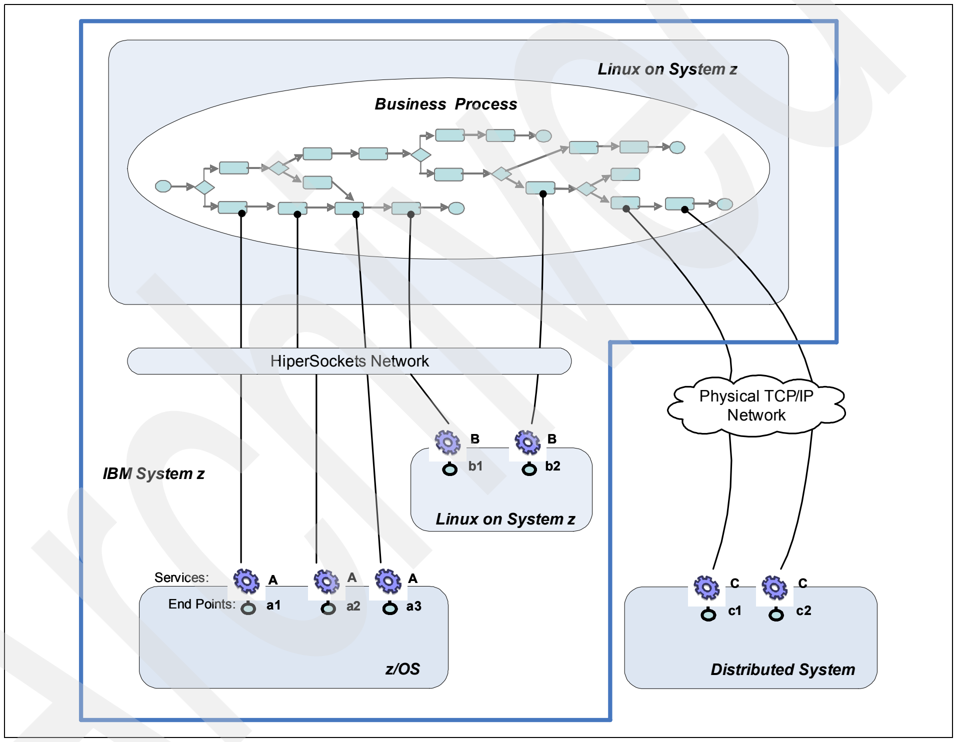


Figure 3-6 Business process services on Linux on System z

Benefits and drawbacks

This pattern benefits from the server virtualization capabilities of Linux on System z as well as the System z hardware capabilities. Virtualization and workload management enable server consolidation on the mainframe by running multiple images on fewer processors. A virtual network within the box using *HiperSockets* will also significantly reduce the network overhead between the business processes on Linux on System z and the business application services they depend on. Frequent calls to services residing on other machines will generally increase network overhead and increase security risks.

3.3.3 Business process services on distributed systems

This pattern describes the business process services residing on distributed systems. All application patterns described in Appendix A, “Business Process services application patterns” on page 481 can apply to this location pattern. In this pattern, both microflows and long-running processes are running on the distributed systems. It could be the same physical machine or, more likely, multiple machines. Figure 3-7 shows the business process residing on the distributed systems. It also consumes some other services on the z/OS system, for example, business application services in CICS and IMS.

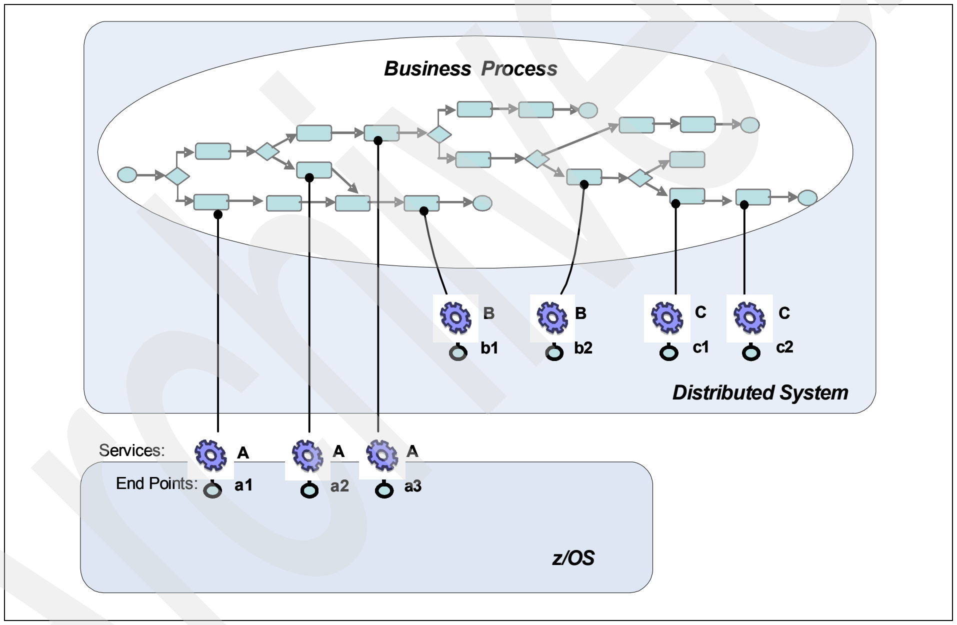


Figure 3-7 Business process services on distributed systems

Benefits and drawbacks

This location pattern is a logical choice if all or most services reside on distributed systems and there is no System z environment. In smaller enterprises and with business processes with light non-functional requirements, using a smaller distributed system might result in a lower cost and easier to run infrastructure. Skills on distributed systems are also widely available.

The issues with this location pattern usually start to occur when the workload starts to scale up heavily and when mission-critical business processes are run. Note that business application services consumed might be mission-critical and therefore the business process itself will also become mission-critical. Running mission-critical business processes on distributed requires multiple distributed servers with a shared database for maintaining process states. The latter is not easy to realize with distributed systems. Also, with this model, the cost of running additional workload on distributed servers goes up linearly. The cost of software licenses is also linear, as are electrical and air conditioning costs. As a result, the cost per unit of work is not reduced as the workload grows.

3.3.4 Business process services on z/OS and distributed systems

This location pattern describes the business process services residing on both z/OS and distributed systems. All application patterns described in Appendix A, “Business Process services application patterns” on page 481 can apply to this location pattern. In this pattern, the microflows are running on distributed systems and the long-running processes are running on the z/OS, or vice versa. In this pattern, the business processes will either start on z/OS or the distributed systems. If the business process starts on z/OS, it becomes the master (control) of the entire process, and the microflow processes on the distributed systems are sub-processes. Conversely, if the distributed systems are the master of the entire process, the microflow processes on z/OS are the sub-processes. In both cases, business application services on either platform can be consumed, including CICS and IMS services. Among the reasons that we might need to link together two or more different business processes could be assets reuse, or mergers of business entities, or an extended business relationship. Figure 3-8 shows the case where distributed systems runs the master process and z/OS runs the sub-processes.

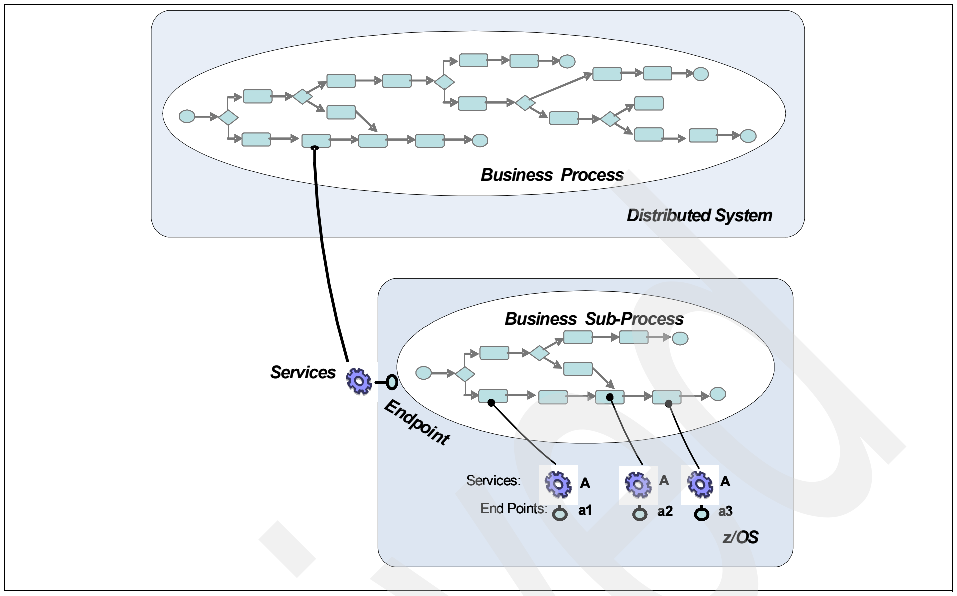


Figure 3-8 Distributed Systems is the master process

In large enterprises with both System z and distributed systems, this location pattern is very likely to be implemented.

Benefits and drawbacks

The benefits of this pattern are very similar to the location pattern with the business processes running on z/OS. However, having a business process server available on distributed systems, business processes exclusively consuming services on distributed systems can be run on distributed. Also, the business process server on z/OS can be kept for business processes with very demanding NFRs. A drawback of this pattern is that skills will need to be kept for running business process servers on both z/OS and distributed systems.

3.3.5 Business process services on Linux on System z and distributed systems

This location pattern describes the business process services residing on both Linux on System z and distributed systems. All application patterns described in Appendix A, “Business Process services application patterns” on page 481 can apply to this location pattern. In this pattern, the microflows are running on the distributed systems and the long-running business processes are running on the Linux on System z, or vice versa. In this pattern, the business processes will either start on Linux on System z or a distributed system. If the business process starts on Linux on System z, it becomes the master (control) of the entire process and the processes on the distributed systems are sub-processes. Conversely, the business process starting on the distributed system will be the master of the entire process and the processes on Linux on System z are sub-processes. Among the reasons we might need to link together two or more different business processes could be to reuse assets, or because of mergers of business entities, or due to an extended business relationship. Figure 3-9 on page 118 shows the case where Linux on System z runs the master process and a distributed system runs the sub-process. In this diagram you also see some services on z/OS being reused. Those services could be, for example, CICS or IMS services.

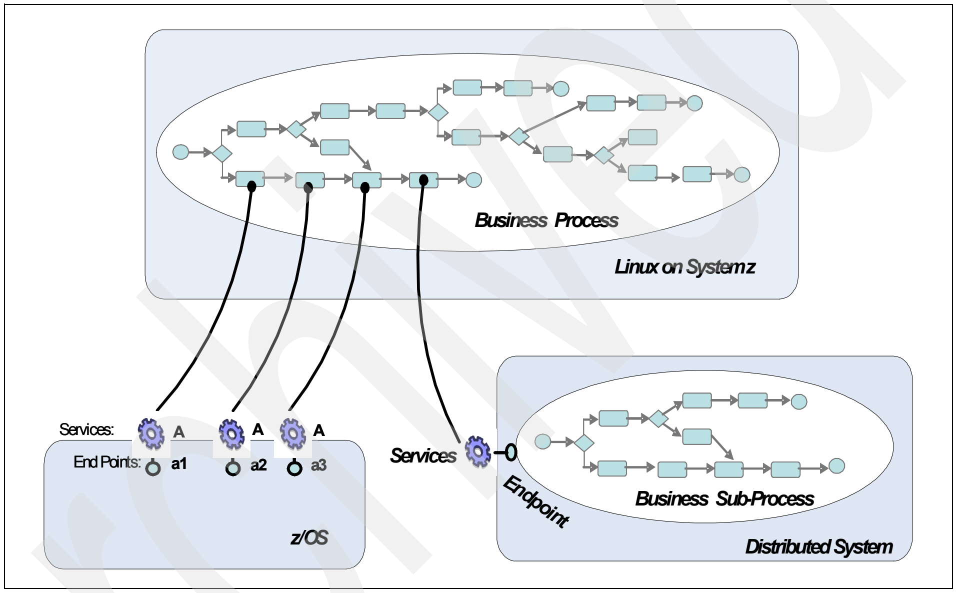


Figure 3-9   Linux on System z is the master process

Benefits and drawbacks

The primary benefit of this scenario is that the business processes running on System z benefit from the server virtualization capabilities of Linux on System z. Additionally, certain business processes can also be run on distributed systems, for example, in the case where most or all consumed services reside on distributed systems too. A drawback again is the maintenance of skills on multiple platforms.

3.4 Platform choice criteria

The next step is to evaluate the platform options based on the needs of the business processes, meaning, decide where to run the business process services solution. When considering server deployment options it is extremely important to consider the characteristics of the business processes being deployed. Some general questions to consider are:

- How does the business process behave?
- What Qualities of Service does the business process require?
- Does the business process need to integrate with any other workloads?
- What data and transactions will the business process access and where do they reside?
- What activity will the business process exhibit, for example, number of transactions per time unit, number of simultaneous users, and so forth?
- How predictable or unpredictable are these activity rates?
- Will the business process require robust backup, recovery, and disaster recovery capabilities?
- Does the business process invoke a large amount of encryption?

These are just some of the questions to be considered when choosing a deployment platform for the business process. In addition, the functionality and features of the server should be considered in terms of how effectively will the server deliver capabilities on behalf of the business process should the business process be deployed on the server. The platform can then be chosen purely based on the requirements gathered during the analysis phase, and aspects such as existing systems and platform investments, customer and developer skills available, and so on, can also been taken into consideration.

A few other considerations for platform choice include, but are not limited to the following:

- If an enterprise already has an investment in a particular server architecture (skills, hardware, software stack), then this platform will generally be the “default” answer. The enterprise’s flexibility to look beyond the default answer needs to be taken into account. Other platform choices would have to provide significant benefits to overcome the inertia of the already installed server architecture.
- Any strategic decision already made by the enterprise mandating a specific server platform for any type of IT will have a strong influence on which platform will be chosen for the business process services. It is important to

keep in mind that a business process server is quite another type of workload and might not fit well on an already chosen server infrastructure.

- The performance characteristics of the proposed business process workloads should be considered for each available platform choice, for example, number of process instances being triggered, maximum number of processes in active state at the same time, volume of data being processed, CPU intensive processes, interfaces with other applications, services and databases, and so on.
- The availability, scalability, and performance characteristics of the server platforms should be considered, taking into account the customer's future plans. The server architecture currently being selected for the business processes must be adequate for at the least the next five years to come.
- The costs of a total solution design, including all environments, not just production, must be considered. The cost of software on the different platforms also needs to be considered.
- The savings due to the ability to virtualize servers should be part of the decision. If there is an expectation that the number of business process servers will be quite dynamic now and in the future, the flexibility of creating and deleting virtual servers becomes important. In some cases there might also be a need to disperse business processes on different servers. This could be the case, for example, when each business unit needs its own server, or in the case of "software as a service."
- In case of migration from another platform, the cost, complexity, risk, and feasibility of any proposed migration has to be evaluated.

Most of these characteristics are part of the non-functional requirements. As with the ESB services, most functional requirements with regard to business process services can be equally met on a number of platforms and as such do not provide a clear basis for platform selection. It is the NFRs that will drive the platform selection. In the next sections, we discuss both the functional and non-functional requirements with regards to business process services.

3.4.1 Satisfying functional requirements

Most of the time, there might not be a clear, direct relationship between functional requirements and the hardware platform; however, the target platform should be able to execute mission-critical processes, provide a flexible infrastructure to enable process changes, reduce process time, and support nondisruptive maintenance; and do all this with reasonable deployment costs. A dynamic and flexible Business Process Engine facilitates the solution building efficiency, reuse of existing assets, and flexibility in changes. The main functions of the Business Process Engine include process automation, choreography, and human-centric support. A typical Business Process Engine supports two different types of business processes: *human workflows* and *automatic business processes*. Business processes allow activities, the steps in a business process, to be performed both sequentially and in parallel. Human workflows are business processes that are driven by humans, that is, people-based activities that are performed on the main path of process execution. The keeping of the process state is important to ensure the process can be continued at a later time or after a failure was recovered. Automatic business processes are run fully automated, that is, as straight through processes without any human involvement on the main path. However, human interaction could be involved for exception handling.

In this book we do not further elaborate on the extended functionality of a business process solution or BPE because there many other sources to learn about functionality. Instead we turn our attention to the non-functional requirements.

3.4.2 Availability and recoverability

Availability is a measure of the time that a system is functioning normally, as well as a measure of the time the recovery process requires after the system fails. In other words, it is the downtime that defines system availability. This downtime includes both planned and unplanned downtime. This means that if we can recover from failures very quickly, we will have a highly available system. For larger production systems, the platform should be able to offer continuous operations. Continuous availability means that high availability and continuous operations are required to eliminate all planned downtime.

Therefore, clustering software uses fault detection mechanisms and automatically fails over the services to a healthy host to minimize the fault detection time and the service recovery time. In an highly available environment it is advisable to use a cluster of business process servers running the business processes, and a separate partition or even machine for the database. To prevent a loss of business access, each hardware and software component required to provide the business process functionality must be protected through at least one level of redundancy. If the business process server software becomes unavailable, the entire business process solution becomes unavailable. It is important to understand that in a mature BPM environment the user communicates primarily with the business process server and not directly with the consumed business application services. It is not unusual that a business process could become the front end of various mission-critical CICS and IMS transactions and database updates on the mainframe, even in a Two-Phase Commit setting. In that case the business process inherits the availability requirements of IMS and CICS.

With regard to availability and recoverability the target platform should fulfill the following criteria for the targeted BPM solution:

- The business processes are capable of dynamically moving from a failed system to another system in a clustered business process server setup. This also applies to business processes in "active" state. Especially in a business process server environment, transactions can be very long. A microflow can be as short as a few milliseconds from start to finish, but a long-running process with various human tasks could be days from start to finish. While a business process is active but waiting for a human task, the database will maintain its state so that at a later point in time the business process can resume again. It is very important to factor in the consequences of any micro flow or long-running business process suddenly breaking because of a BPE, LPAR or machine failure. Business processes driving multiple database updates might fall under a Two-Phase Commit scope and in case of an outage there must be a guarantee that either all updates are made or all updates are rolled back or compensated.
- Comprehensive design for availability is required when *continuous operation,* such as 24 x 7 x 365 availability is expected. For example:
  - A 99% system availability indicates 3.7 days of unavailable time per year, which is unacceptable in some environments (ATMs, Internet banking, airline reservation systems, and so on).
  - A 99.99% of system availability indicates 52.5 minutes unavailable time per year, which is realistically acceptable for almost all administrative business processes. However, for systems such as the ones being used in nuclear power plants or in airplanes, this availability is unacceptable. Also, just one outage of 52.5 minutes a year in, for example, an airline reservation system is unacceptable too. On the other hand, ten times a year an outage of 5.25 minutes might be OK.
  - A 99.999% system availability indicates a 5.3 minutes unavailable time per year.

  Not surprisingly, availability requirements have to be formulated very precisely and not just by requesting the same percentage (many times too high) for all business processes.
- The business process solution must be capable of routing workload among multiple servers, eventually even splitting up a long-running process among multiple servers. Queues must exist to hold requests until a server becomes available to continue processing. Please note that we are not referring to WebSphere MQ here, but to a workload management queuing system, which is usually supplied by the operating system.
- The business process server solution must be able to prioritize work. Higher priority work should be processed before lower priority work in case of limited

system resources. At the same time, balancing of workload and anticipating near future decreases or increases should be possible.

- Ideally, the business process solution can start and stop server processes as needed. This will benefit vertical scalability of the solution.
- If a server fails, the failed business processed must be picked up by a backup of the same business process or automatically restarted.
- A failure of one image in the cluster must not affect the other images, and any specific business process transactions running on it can be fully dispatched and recovered in any other image, making use of an architecture that provides (process state) data sharing.
- Business processes with high business resiliency and disaster recovery requirements might need to be supported. In each country, government rules might exist for DR in a certain industry, such as banking. The business process solution might need to be capable of recovering all business processes on another system at a significant distance in case of a datacenter outage. DR architectures are not uncommon and most enterprises have them, but they might not have factored in the typical characteristics of business processes, which could be like transactions, but with a very long time, making the data sharing aspect even more important.

3.4.3 Reliability

Reliability is the experience of the given platform to sustain acceptable service levels and with high data quality. For example, an extremely high reliability could mean an individual server providing multiple quarters without unplanned system outages. To prevent a loss of business process services, the hardware and software component that is required to provide the business process functionality must be protected for any unexpected system outage.

Data integrity and reliability is critical to the business process. Process states need to be kept during the process execution path. A reliable Database Management System (DBMS) is critical to the business process services solution.

3.4.4 Performance and capacity

In general, there are two different business processes: long-running processes (sometimes referred to as macro flows) and short-running processes (microflows).

Long-running processes are interruptible, for example, by a human task or a complex business-to-business interaction with other stateful services. Each activity in the process can run in its own physical J2EE transaction. One transaction encloses one or many steps in the business process. The J2EE transactions are chained using persistent messaging. The transaction boundaries can be optimized by the developer of the business process. After each transaction the state of the business process is written into a datastore. Long-running processes survive system failures and support parallel execution of activities that are on parallel paths within a business process.

Microflows run within one J2EE transaction on one thread from start to end without interruptions. They are light weight compared to long-running processes because they do not need any persistent intermediate state to be written to a datastore, nor do they send any persistent messages while navigating.

The server used must be well balanced. A server with one (or many) very fast CPUs but low memory or low I/O performance will have negative impact on the long-running processes. For long-running processes, high I/O performance in the form of multiple, fast disk drives is as important as enough processing power and a sufficient amount of memory. As a general rule, the machine running the database management system for the Business Process Engine will require fast and many CPUs, large CPU caches, a large amount of physical memory, and good I/O performance.

CPU usage could be very high when there is a high amount of XML messages to be processed.

The target platform should fulfill the following performance criteria for the targeted BPM solution:

- Ability to handle sustained peak workload utilization of 100% without service level degradation.
- Scalable and high performing connectivity to the enterprise data and applications.
- High transaction throughput for the business processes (Business Transaction Per Second or BTPR).
- Ability to assign an appropriate workload profile for the server (or workload services class) for different workload requirements.
- Ability to run the process server and the database on the same physical machine; fast CPUs, enough memory, and the speed of disk I/O are important.
- Fast CPUs are also important when there is a high amount of XML messages being processed.

- A well performing messaging mechanism (engine) is required due to the process engine's internal messaging requirements. This is especially important if JMS is a requirement for the business processes.
- Robust queue management and high I/O bandwidth for data movement for deployment of the master process flow.

3.4.5 Transactionality

One of the main differences between long-running processes and microflows is their transactional characteristics:

- A *microflow* is executed within a single unit of work. The unit of work might either be a global transaction, or an activity session.

  Figure 3-10 on page 125 shows the transaction boundaries in a microflow situation.

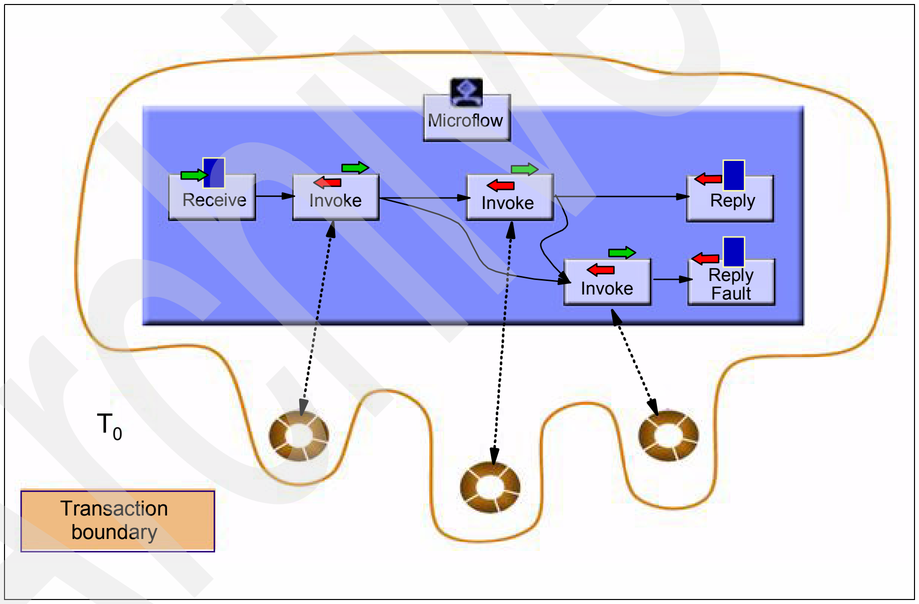


Figure 3-10 Short-running transactions (microflows)

- A *long-running process* comprises multiple global transactions. Each navigation step of a long-running process is performed in its own transaction. A navigation step might span multiple activities.

  Figure 3-11 shows this situation.

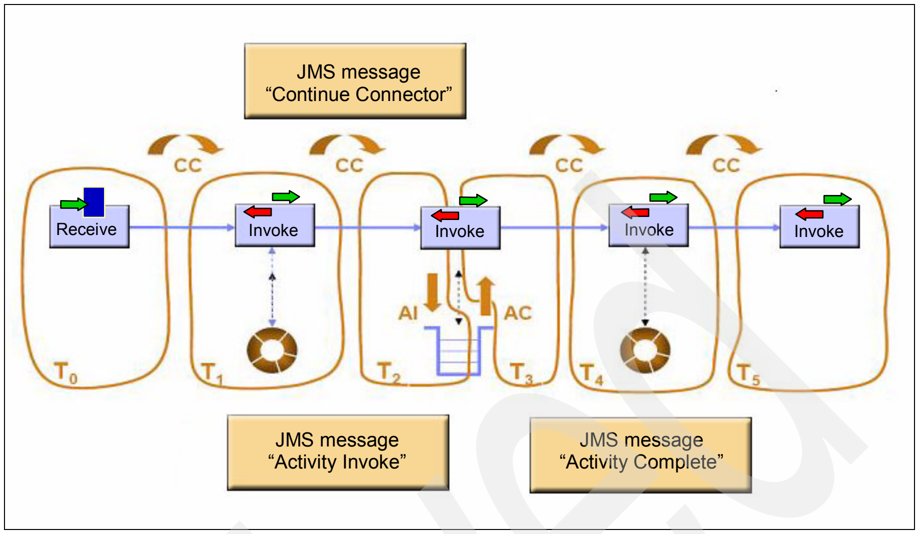

JMS message "Continue Connector"
CC
CC
CC
CC
Receive
Invoke
Invoke
Invoke
Invoke
AI
AC
T_0
T_1
T_2
T_3
T_4
T_5
JMS message "Activity Invoke"
JMS message "Activity Complete"

Figure 3-11 Long-running processes and transactions

Looking at a long-running process, its navigation is performed as a series of transactions. Each single transaction is triggered either by an external request, an internal continuation message, or a response from an asynchronous service. The transaction's execution comprises the navigation of one or more activities of the process, including the respective process state changes in the state database. The last step of such a transaction might be the sending of a continuation message to ensure follow-on navigation of the process.

Long-running processes are forward recoverable. If there is a failure, the transaction that was in flight rolls back and is retried, causing navigation to continue from there. Navigation state is kept in the runtime database in the form of process instance state. In addition, the messaging system holds navigation state in the form of external requests, responses, and internal continuation messages to drive the navigation of the process.

In general, the database, the messaging system, and transactional resources used by invoked services all participate in a two-phase commit protocol. As long as a transaction has not been committed, none of the changes done on its behalf are visible to the outside world. The isolation property guarantees that they only become visible when the transaction's final state has been reached. Therefore, the target platform should support transactional integrity with system-wide *Two-Phase Commit*.

This works well for short-running processes that invoke synchronous operations. However, when a process involves asynchronous steps, for example, steps involving human interaction, intermediate results of the process must be made visible. A compensation logic (reverse the update) needs to be added into the business process in order to handle this kind of situation, but this is outside the scope of this topic and is not be discussed in detail in this book.

3.4.6 Scalability

System scalability requirements need to be considered in the hardware choice. Some of the factors to include are anticipated growth in the amount of business processes to run, the timeline for this growth, and if the business processes are architected for a scale up or scale out architecture. It is also important to be able to handle a business process with an unpredicted growth. The system could require additional capacity to be added dynamically. The ability to dynamically add, remove, and redeploy resources is key in optimizing the server's utilization and maximizing the availability of the platform and supported applications. The platform of choice should fulfill the following scalability criteria for the targeted BPM solution:

- The number of business process engines and servers can be dynamically increased or decreased based on the workload. Business process engines and servers can be scaled up or down vertically and horizontally. The target platform can be a cluster (for example, of up to 32 images in different partitions on different machines and possibly geographically dispersed) with full data sharing and high availability and recovery.
- Resources, such as CPU and memory, must be added dynamically when there is a sudden increased workload.

On the software side, the process server should be able to scale automatically with any unpredicted workload growth.

3.4.7 Security

Enterprise security has become a key business requirement. The security strategy for BMP should include key elements such as having a secure platform infrastructure, ensuring data privacy and compliance and audit, and management of security across the entire business processes.

Security criteria for the targeted BPM solution include the following considerations:

- Centralized security functions such as user identification and authentication are critical. Business process users and administrators should be authenticated and authorized for accessing the process instance and resources. Also, access via Web services should be authenticated. Therefore, the target platform should be able to provide the required mechanism and repository. Authentication services are critical to process server functionality. Without the authentication services, users cannot gain access to the process services.

- Resource access control and auditing is required for both the operating system and business processes running on the system. The target platform should provide mechanisms to prohibit any unauthorized resources access and logging should occur.
- Audit data is essential for ensuring that the customer's installation security policy is being followed. Typical business events undergo a complex series of interactions such as business rules, flows and service calls, and those events probably happen across different servers and in different time zones. Some events are long running and some are short running. An essential logging infrastructure is important.
- Encryption might be required if the business processes are exposed to or require communication with other Web services. The target platform should provide adequate support for encryption at both the software and hardware levels. A minimum is a key management system and a cryptographic processor. The target platform should provide strong cryptographic support to handle large volumes of encryption and provide the methodology to help protect and manage keys.

3.4.8 Accounting

One of the main goals of SOA is reuse. In a successful SOA, business processes and their services increasingly will be reused over time by both internal and external users. An example is a business process for claims processing implemented at a large insurance company and used by a large number of insurance agents. Because in an SOA components are shared and not owned by individual business units, a good mechanism for charge back will become essential in most situations. Charging an entire business process or certain steps of a business process to somebody, either the user or the owner, is a quite common requirement. The platform of choice will need to support this, ideally with built-in software. To be able to charge back, it is necessary to know who to charge to. If this will be the end user, then a proper and reliable mechanism for authentication is required.

The most important platform requirements for accounting and charge back purposes are the following:

- Accountability required by a combination of user authentication and the ability to propagate a user’s credentials throughout the application. The information can be used for charge back to the process owner or the end user.

- The ability to collect all accounting and performance data and present this back for reporting. These records can be analyzed and aggregated in performance and capacity reports, but can also be used as a base for security analysis and accounting policies.

3.4.9 Virtualization

A business process services solution usually runs in a multi-tier environment, but the target platform should support virtualization, including virtualized IP network, virtualized hardware resources, virtualized servers, and so forth.

3.4.10 Manageability, usability, maintainability

The platform's manageability and usability might not be related to the business process solution, but there are a few other things regarding the target platform that we should consider:

- The target platform should be designed and able to manage multiple business process workloads in a single system image or in a set of clustered images. The target platform should be able to support mixed workloads, that is, the business process solution running simultaneously with any other existing business applications on the same image or the same machine, while guaranteeing the service levels agreed for each business process.
- The target platform should have a rich tool set for managing, supporting, and controlling complex simultaneously running targeted business processes and their components.
- The availability of the platform skills is also significant. Supporting skills and resources to manage the targeted platform and the new business process solution are necessary.
- The target platform needs to support the open standards and the industry standards for the business process solution.
- The target platform must permit maintenance to be applied concurrently with production operation by allowing individual business process engines to be removed and upgraded or replaced without interruption of service for the end user.

3.5 Business process services solutions

The next step after choosing a runtime pattern and platform is to determine the product to be used. In the previous section we discussed several criteria to inform the choice of platform with respect to the business process engine or server. Selecting a software product that fits the customer's requirements so that the solution can grow along with the on demand business is the next step. This section discusses the possible business process services software solutions that IBM offers on System z.

3.5.1 Business process runtime pattern product mapping

In this section we select the *Exposed Serial Workflow runtime pattern* and map it to IBM products. In this mapping, the Business Application services at the enterprise secure zone are implemented by WebSphere Application Server, CICS Web Services, and so on. The ESB and the Exposed ESB gateway are implemented by WebSphere Message Broker or WebSphere ESB. Business Application services at the partner site invoke the automated process instance implemented by WebSphere Process Server over the Internet using SOAP/HTTPS. The IBM HTTP Server acts as a connector by exposing SOAP/HTTPS to partner organizations while allowing the WebSphere Process Server to process standard SOAP over HTTP (SOAP/HTTP) calls. The Service Integration Bus within WebSphere Process Server is configured to secure all transactions to the external Partner Zone using WS-Security integrity and confidentiality. This is illustrated in Figure 3-12 on page 130.

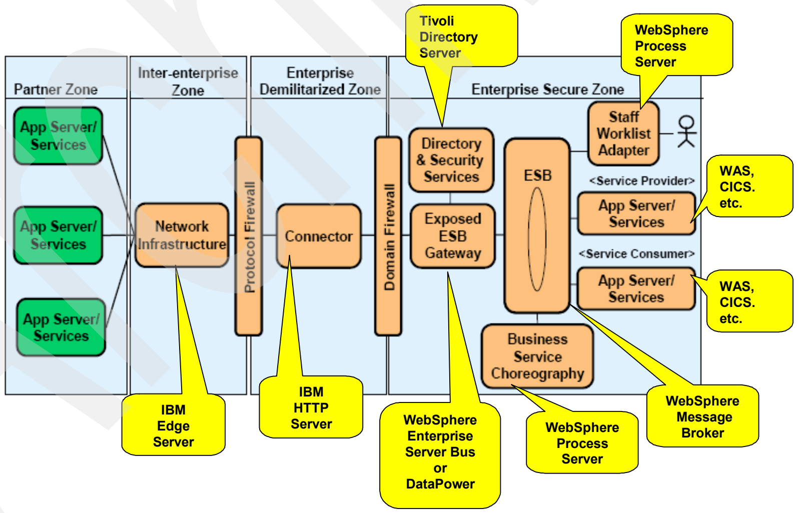


Figure 3-12 Exposed Serial Workflow runtime pattern: WebSphere Process Server product mapping

There are other process server products available from other software vendors, such as Oracle BPEL Manager, SAP NetWeaver® Application Server, BEA AquaLogic BPM and WebLogic Integration, TIBCO BusinessWorks, and Microsoft BizTalk® Server. We do not discuss those products at this time.

3.5.2 IBM's BPM product suite

Today, no BPM offering exemplifies the cycle from modeling through to performance management, all layered on SOA and Internet standards, better than IBM's WebSphere BPM suite. The IBM WebSphere BPM product suite has four primary components, each playing its role in the business process life cycle:

- Model stage
  WebSphere Business Modeler provides process modeling, KPI definition, and optimization through simulation analysis. For more information, refer to:
  `http://www.ibm.com/software/integration/wbimodeler/`
- Assemble stage
  WebSphere Integration Developer (WID) provides process design and development using component assembly. It is an IT tool, but it does not necessarily require programming. Skeleton assemblies and data definitions can be imported from WebSphere Business Modeler to create a clean business-IT handoff. For more information, refer to:
  `http://www-01.ibm.com/software/integration/wid/`
- Deploy stage
  WebSphere Process Server (WPS) is IBM's business process engine. WebSphere Process Server is essentially the SOA hub of the WebSphere Application Server, based on J2EE and Web Services standards. Assemblies created in WID are executed on Process Server, automating human workflow and integrating diverse business systems via SOA middleware. WebSphere Process Server includes the *WebSphere Enterprise Service Bus (ESB)*, a secure standards-based communications fabric linking all process components and capable of invoking IT services across the enterprise. For more information refer to:
  `http://www.ibm.com/software/integration/wps/`
- Manage stage
  WebSphere Business Monitor provides business process performance management. It captures, filters, and aggregates events from the WebSphere Process Server and continuously updates the KPIs previously defined in WebSphere Business Modeler. It displays performance metrics in graphical management dashboards running on WebSphere Portal and accessed through a Web browser, supporting both strategic and operational views of business process performance data. WebSphere Business Monitor also provides *Business Activity Monitor (BAM)*, rule-based monitoring of KPIs

with automatically triggered actions. Because WebSphere Business Monitor was designed to work with WebSphere Business Modeler, actual performance results can be fed back easily to update modeling assumptions, improving simulations and accelerating the cycle of continuous process improvement. For more information, refer to:
`http://www-01.ibm.com/software/integration/wbimonitor/`

Several other IBM products can be part of and also be used for Business Process Management, including:

- Model stage
  IBM Rational® Software Architect (RSA)
- Deploy stage
  IBM WebSphere Adapters and IBM WebSphere Portal
- Manage stage
  IBM Tivoli Access Manager (TAM), IBM Tivoli Federated Identity Manager (TFIM) and IBM Tivoli Composite Application Manager for SOA (ITCAM-SOA)
- Governance
  IBM WebSphere Service Registry and Repository (WSRR)

IBM WebSphere Portal could also be used to provide (deploy) a Portal front end for users completing human tasks within a business process running in IBM WebSphere Process Server.

The key WebSphere products are mapped to the BPM life cycle in Figure 3-13.

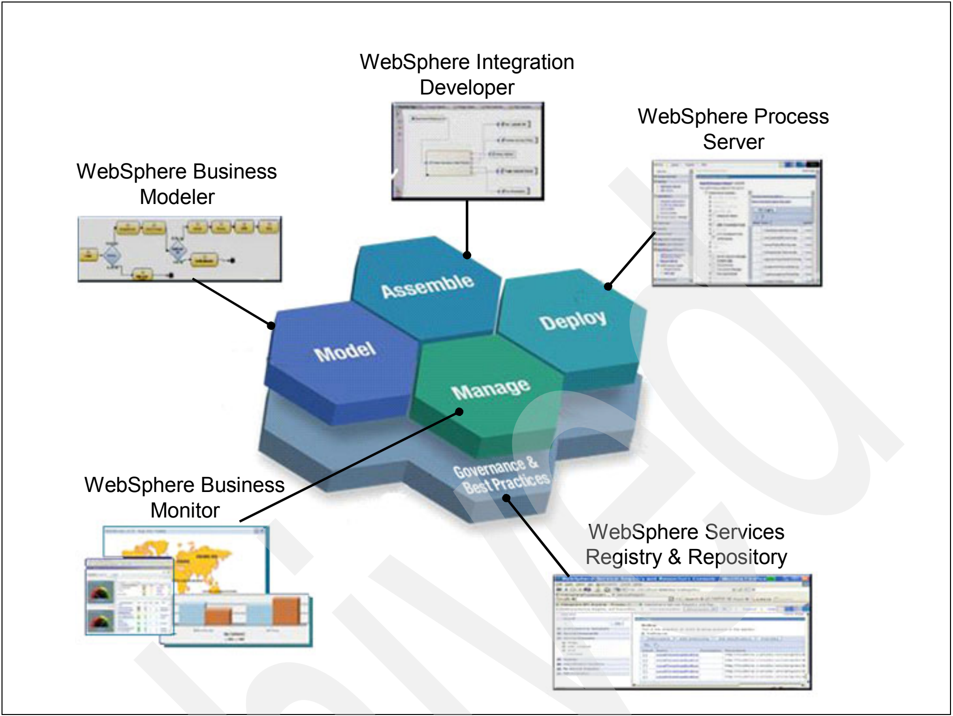

WebSphere Integration Developer
WebSphere Process Server
WebSphere Business Modeler
Assemble
Deploy
Model
Manage
Governance & Best Practices
WebSphere Business Monitor
WebSphere Services Registry & Repository

Figure 3-13 IBM WebSphere BPM product mapping in BPM life cycle

Of the four core IBM BPM products, IBM WebSphere Business Modeler and IBM WebSphere Integration Developer are Eclipse-based desktop tools that usually are run on a Windows platform.

WebSphere Business Monitor has two main components: the *Monitor Server* and the *Dashboard*. The Monitor Server is usually installed and run on the same server as the WebSphere Process Server; the Dashboard is installed and run on the Portal Server. We do not further discuss the tools in this section, but focus on WebSphere Process Server running on System z. The WebSphere Services Registry and Repository (WSRR) product is discussed in Chapter 2, "Enterprise Service Bus services" on page 5.

3.5.3 IBM WebSphere Process Server

IBM WebSphere Process Server is based on WebSphere Application Server and includes many capabilities such as business process automation, business state management, business rule management, human task management, and advanced transformation. There are many features and capabilities available in WebSphere Application Server that are inherited by WebSphere Process Server, such as performance optimization, scalability, security, high availability, transaction management, and basic resource management. All additional Quality of Service features offered by WebSphere Application Server for z/OS are implicitly available to WebSphere Process Server on z/OS too.

WebSphere Process Server runtime architecture

As shown in Figure 3-14, the WebSphere Process Server runtime architecture includes three layers:

- SOA core
- Supporting services
- Service components

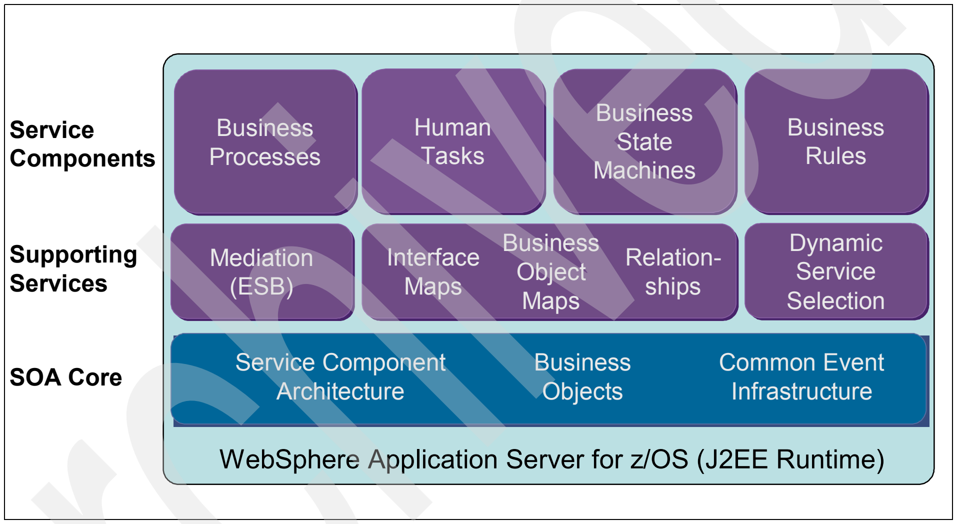


Figure 3-14 WebSphere Process Server runtime architecture

The *service-oriented architecture (SOA) core* is the foundation in WebSphere Process Server. The main components of the SOA runtime are:

- Service Component Architecture (SCA): The uniform programming and invocation model for business services that publish or operate on business data.
- Business Objects (BOs): Represent the data that is passed within the framework.
- Common Event Infrastructure (CEI): Provides the foundation architecture for the Business Process Management (BPM) and Business Activity Monitoring (BAM).

Supporting services provide the transformation primitives for the WebSphere Process Server service components. This layer provides a common mapping and transformation service to enable mapping of interface signatures via the interface map, and correlation and synchronization of key relationship interfaces between systems with the relationship service. The selector component provides for dynamic component or target invocation with flexible administration control at runtime.

Service components are the high-level components of WebSphere Process Server and can be connected together to create powerful solutions based on the SOA core technology. As a result, the service components provide the integration implementations and act as both service providers and service consumers as part of an SOA-based approach to integration through standard SOA relationships.

For more details of these components, refer to the IBM WebSphere Process Server product-specific documentation.

WebSphere Process Server runtime components

Figure 3-15 on page 136 shows the major components of the WebSphere Process Server (WPS). Several components (such as Load Balancer, IBM HTTP Server (IHS), LDAP, and Database Server) are not shipped with the IBM WebSphere Process Server product, but are included in the diagram to show the complete context of the runtime environment.

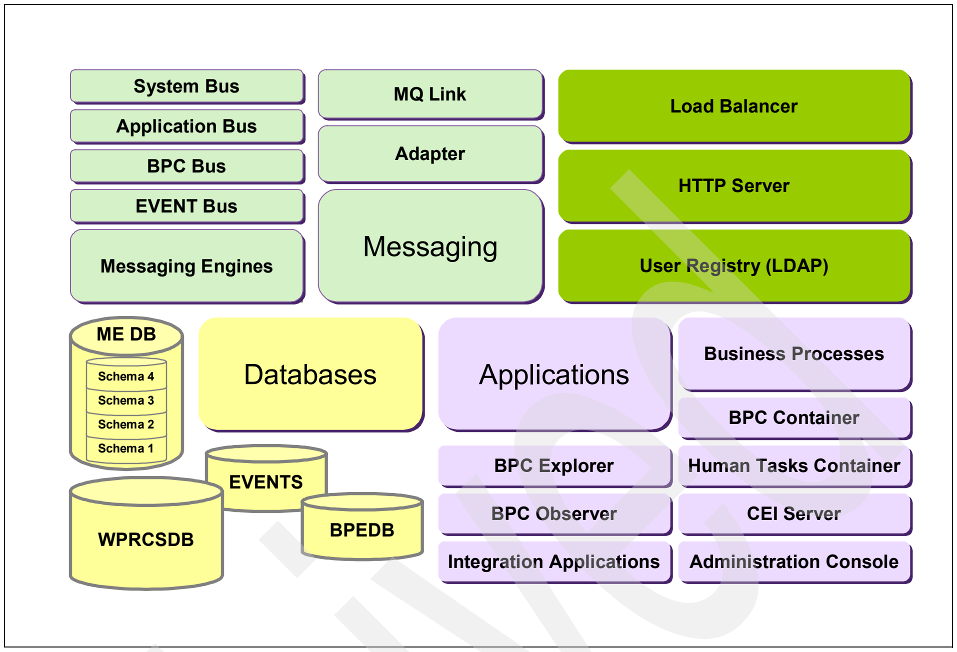


Figure 3-15 WebSphere Process Server runtime components

See the IBM WebSphere Business Process Management Information Center at `http://www.ibm.com/software/integration/wps/library/` for detailed descriptions of each IBM WebSphere Process Server components.

Benefits of IBM WebSphere Process Server on System z

WebSphere Process Server for System z leverages all the System z features, such as the zAAP Specialty Engine, z/OS Parallel Sysplex, WLM, Cryptographic processors, HiperSockets, RACF, and so forth. WebSphere Process Server for Linux on System z also leverages the IBM System z features, including IFL Specialty Engine, Cryptographic processors, HiperSocket, virtualization, and so on.

Following are several key benefits of running WebSphere Process Server on System z:

- Use the exceptional workload management capabilities of System z, including:
  - PR/SM and IRD on both z/OS and Linux on System z
  - On z/OS only, WLM and z/OS Parallel Sysplex technology, whereby z/OS WLM actively manages the work against customer-specified goals, using CPU, memory, and I/O prioritization across the sysplex

  - On z/OS only, achieve scalability and durability through z/OS Parallel Sysplex features and multi-process design of WebSphere Application Server for z/OS

  The benefits of previously mentioned technologies become really clear if there is a requirement for a high availability heavy-duty business process server implementation with challenging SLAs. With such a requirement, the business process server will need to be implemented across multiple images, servers, and even sites, to guarantee sufficient availability. In such an implementation, workload management is provided as follows:

  - IRD balances resources, such as CPU
  - WLM balances business process requests across available servers and images, based on workload and system usage
  - Parallel Sysplex provides the infrastructure to let requests travel freely between different LPARs and machines
  - GDPS can be optionally used for disaster recovery in case of a system or site outage

- Benefit from HA and scalability infrastructure, such as Parallel Sysplex, Sysplex Distributor, DVIPA, GDPS, and so on.

  WebSphere Process Server on z/OS runs on top of WebSphere Application Server for z/OS and can thus be set up in a clustered environment across multiple z/OS LPARS. Using Parallel Sysplex and DB2 data sharing, Business Process Engines will have access to one data store for maintaining state. This means that in case of an outage of an LPAR or a BPE on a LPAR, another BPE on another LPAR can continue processing without interrupts and based on the exact same data image.

- Benefit from “features” in the System z pricing model. For instance, the substantially discounted prices of the IFL engines deliver performance that is equivalent to the general CPs. Also, WebSphere Process Server is a heavy Java and XML user. Java processing can be offloaded to specially priced zAAP engines and XML parsing can be offloaded to specially priced zIIP engines[1].

  WebSphere software on z/OS might qualify for sub-capacity WLC pricing and significant savings can be achieved. Refer to the following Web site for details: `http://www.ibm.com/servers/eserver/zseries/swprice/`

  WebSphere software on Linux on System z can be used in combination with IFL licenses.

- If most data and back-end services run on System z, and particularly z/OS, it is a benefit to run the BPM solution as close as possible to that data and

[1] Offloading XML parsing to zIIP engines requires certain z/OS levels. For details refer to: http://www-03.ibm.com/systems/z/advantages/ziip/index.html

back-end services. Co-location, as stated earlier in this book, means less network overhead, reduced complexity of the IT infrastructure, and fewer security exposures.

  If the BPM solution runs on Linux on System z and data and business services are mostly on z/OS, then HiperSockets can be used to communicate quickly, securely, and efficiently between the Linux on System z partitions and the z/OS partitions.

- Companies already running a System z have well-established tools and procedures for managing and operating System z with a minimum amount of personnel while guaranteeing extreme SLAs. Running the BPM solution on System z means continuation of that regime for the BPM solution too. Business processes will need to be managed as professionally as the back-end services those business processes access.

Most other System z benefits discussed elsewhere in this book also apply to the business process server running on System z.

3.5.4 Business process services solution mapped to System z

This section describes the business process services solution architecture on the IBM System z environment. The solution takes into consideration cost effectiveness, workload balancing, scalability, high availability, reliability, maintainability, and performance. In many cases, the core business applications run on the z/OS platform because of requirements for the level of QoS that System z provides. In many cases it makes sense to run the Business Process Engine component on the same platform where the business application services it depends on are also deployed.

Figure 3-16 on page 139 shows a platform-independent view of a business process services solution. The text that follows describes how this solution would look on System z.

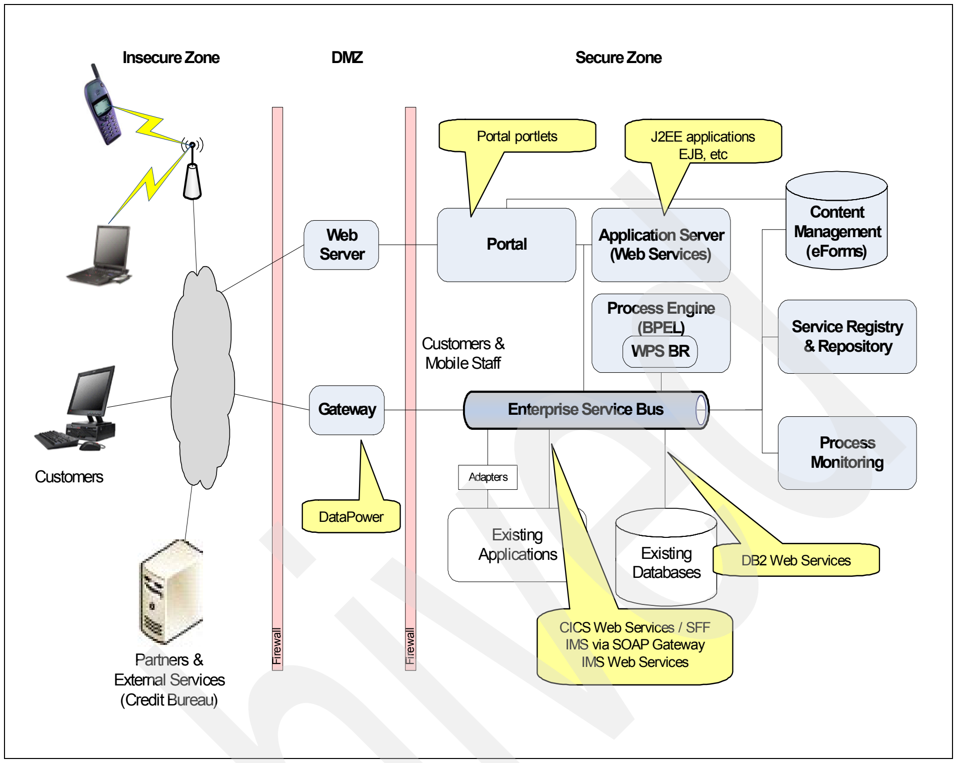


Figure 3-16 Business process services solution: Simplified view

This business process solution represents an integration of the various other services provided by the SOA. The integration characteristics are:

- The Business Process Engine runs inside the IBM WebSphere Process Server, and business processes are exposed by the ESB infrastructure as Web services.
- Enterprise J2EE applications are running in IBM WebSphere Application Server.
- Enterprise CICS applications are Web Services enabled.
- Enterprise IMS applications and databases are Web Services enabled.
- Enterprise DB2 database are Web Services enabled.
- Enterprise Portal applications with business portlets are running in IBM WebSphere Portal Server.
- All the services mentioned are exposed through the ESB infrastructure as Web services.

Business Process Engine using WPS

The key product in the WebSphere suite of products is the IBM WebSphere Process Server (WPS), which allows users to build sophisticated composite applications. WPS is a single integrated run time for service-oriented integration. The high performance Process Engine can run the critical business processes securely, consistently, and with transactional integrity. WPS is also an integrated run time for all SOA-based process automation. WPS is a run time Process Engine for all the components defined in the Assemble phase (WS-BPEL, State Machines, Business Rule, and so forth). See Figure 3-17 for a WPS integration view.

Reasons to run the Business Process Engine on System z are discussed in "Benefits of IBM WebSphere Process Server on System z" on page 136. See "IBM WebSphere Process Server deployment configurations" on page 153 for deployment configurations on Linux on System z and z/OS.

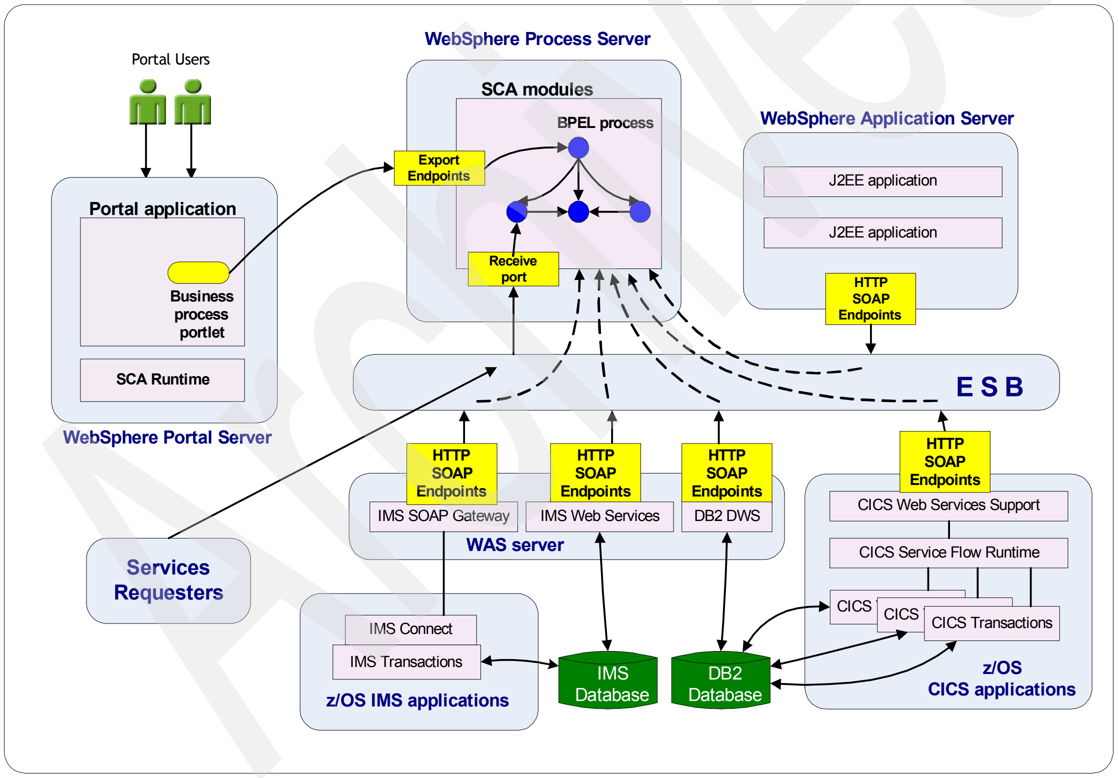


Figure 3-17 WebSphere Process Server integration

Using ESB as the Business Process Engine

It is possible to implement Business Process Services in the ESB, but only those that are microflows. Business state machines, business rule engines, and human tasks are not provided by the ESB. In other words, the ESB can be used as the process engine for simple workflows as long as there are no human interventions or state machine requirements. Instead, we use ESB as the routing and transformation basis for the business process integration.

ESB integration to WPS

The ESB comprises a set of interaction points to which services are connected. Services are able to invoke other services that are connected to the bus. The ESB allows the creation of dynamic and flexible connectivity between services during service invocation without changing service consumers or providers. Interactions can be managed and measured. It simplifies an SOA by minimizing the explicit connectivity between services. Business processes can be exposed as services by the ESB. It is more logical to run the business process solution on the same platform where the services it depends on are also deployed. The SCA components of the IBM WebSphere Process Server will communicate with the SOA-enabled services reachable via the ESB.

IBM ESB product offerings support and implement the enterprise ESB infrastructure using two major products: *WebSphere Enterprise Services Bus (WESB)* and *WebSphere Message Broker (WMB)*. Figure 3-18 shows a high level view of exposing the business process as Web services by the ESB. We also extended the ESB to the insecure DMZ using the DataPower XML appliance.

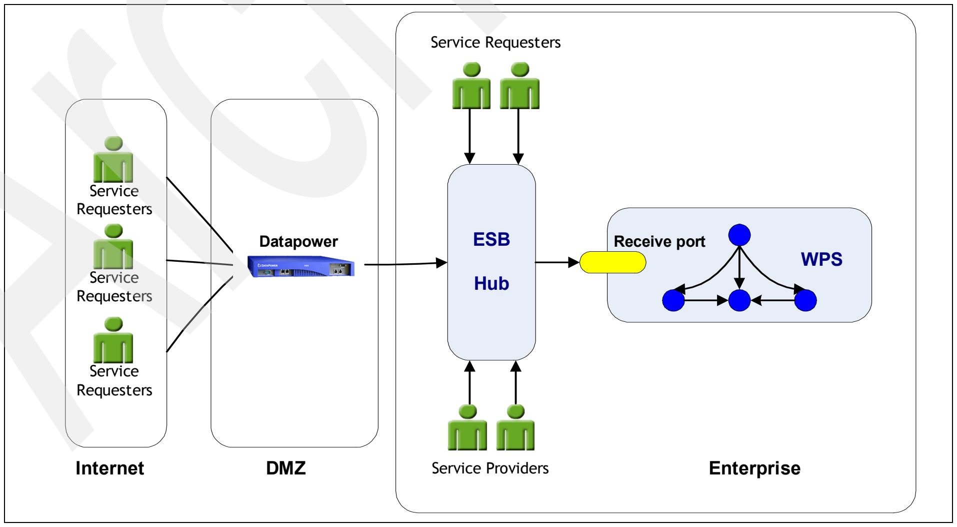


Figure 3-18 Business process exposed as Web services

Using a J2EE application server as the Business Process Engine

Business process services can be implemented on a J2EE application server such as IBM WebSphere Application Server. In this scenario, you need to develop your own Business Process Engine logic for microflows and long-running processes. You also need to develop your own Human Tasks, Business State machine, and Business Rule engine logic. However, the alternatives would be buying a vendor business engine solution that is capable of deploying on a J2EE application server.

Enterprise J2EE application integration to WPS

IBM WebSphere Application Server delivers flexible, secure infrastructure to provide a reliable foundation for your service-oriented architecture. We integrate existing Web-services-enabled WebSphere Application Server applications into WPS. It makes sense to run the Business Process Engine on the same platform where the WebSphere Application Server runtime environment already exists.

Figure 3-19 on page 143 shows a simplified view of some enterprise Web services on z/OS that are exposed to the ESB and orchestrated by the WPS BPEL workflow.

There are many different paths to use in the generation of a J2EE Web service. We identify three principal approaches here, determined by the elements used to start the creation of the service. A Web service can be implemented from:

- An existing application (bottom-up): Transforming an existing application into a Web service includes the generation of service wrappers to expose the business functionality.
- An existing service definition, WSDL, to generate a new application (top-down): Generating a new application includes using a specific programming language and model.
- An existing group of already generated Web services to provide a new combination of functionality (multiple services): Composing a new Web service might include the use of workflow technologies.

There are many other IBM documents that cover Web services development in detail, so we do not discuss this any further in this book.

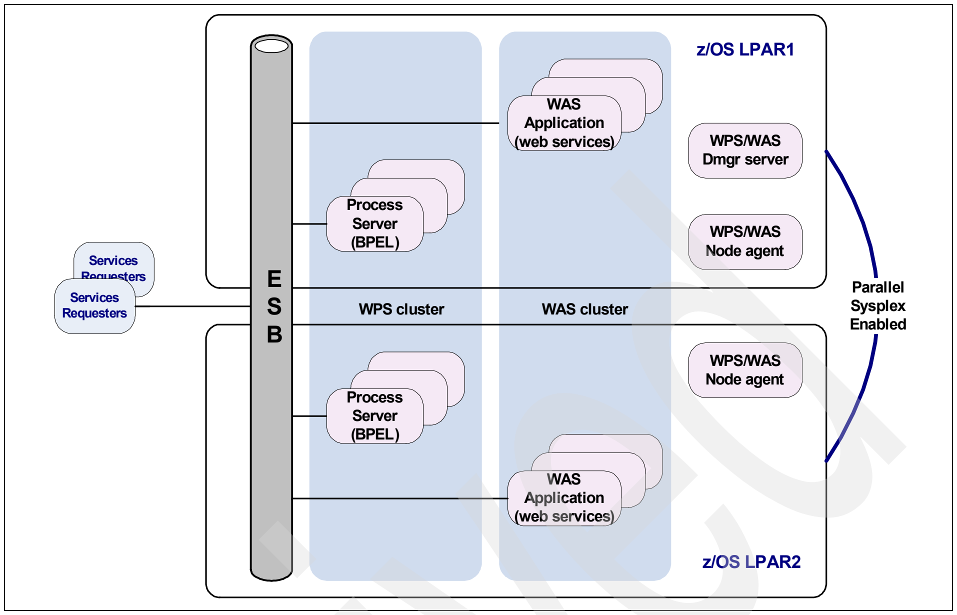


Figure 3-19 Enterprise Web services - J2EE applications on z/OS

Using CICS SFF as the Business Process Engine

The *CICS Service Flow Feature (SFF)* provided by CICS TS Version 3.1 and later can be used to create Web services from existing CICS 3270 screens and program-oriented application assets. CICS SFF provides the capability to implement CICS Business Process services by composing a sequence of CICS application interactions. CICS SFF composes fine-grained applications into business-level service components. Therefore, all these business services components built using the CICS Service Flow Feature can be used to implement straight-through business processes such as microflows. However, long-running processes, people interactions, business rules engines and business state machines need to be handled by the CICS applications, and this could be very complex.

Enterprise CICS applications integration to WPS

The latest versions of CICS TS also support Web Services. Web Services has become an integral part of CICS and provides support for standards such as the SOAP protocol and WS-AtomicTransaction. With CICS Web Services support, CICS has the ability to act as a service provider and a service requestor, and it can fully participate in a modern standards-based SOA.

The CICS Service Flow Feature can also be used to create Web services from existing CICS 3270 screens and program-oriented application assets. The CICS

SFF provides the capability to implement CICS business services by composing a sequence of CICS application interactions. The Web services created can then be exposed on the ESB and consumed by the WPS as a part of the business process.

Figure 3-20 shows an example of exposing a CICS application as a Web service using CICS Service Flow Feature and CICS Web Services Support and consuming this Web service by a requester. In this scenario, the Web service requester is WPS business process services.

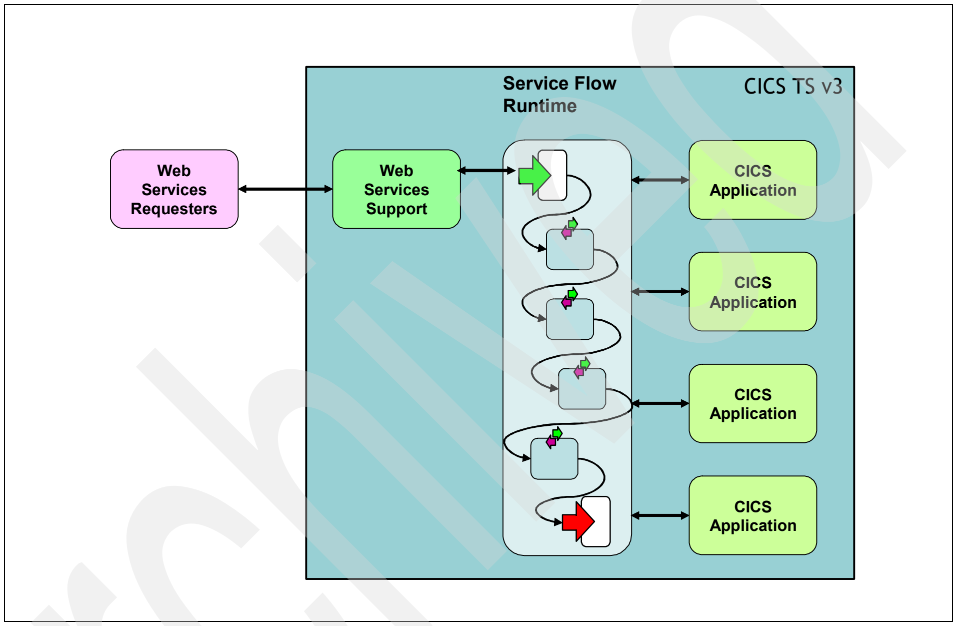
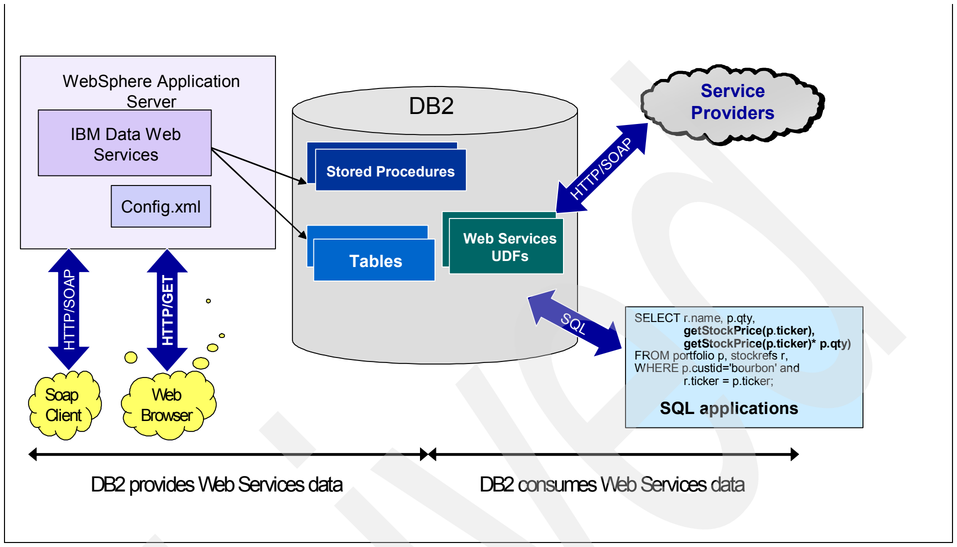


Figure 3-20 CICS Web Services and Service Flow Feature

Enterprise DB2 data access via Web Services

Although DB2 cannot be a Business Process Engine, we can integrate DB2 into a Business Process. Getting the right data quickly and consistently for all applications continues to be a key challenge for enterprises. Information as a Service is key, that is, information that SOA can exploit. With the new generation of DB2 Web Services tooling, called *Data Web Services (DWS)*, simple and fast SOA enablement of DB2 database operations can be achieved. DB2 can now provide Web Services based on data and stored procedures and can also consume Web Services data. DWS is the successor of *Web Services Object Runtime Framework (WORF)* and is fully re-architected with more focus on logging, tracing and monitoring, better tool integration, full XML integration, XSLT processing, and better Web 2.0 integration. DWS also makes more use of standard Java libraries. Figure 3-21 shows a high level view of accessing DB2 for z/OS data in SOA. Therefore, with DWS, database data can be exposed as a service and it can be integrated into the Business Process solution.



```
SELECT r.name, p.qty,
        getStockPrice(p.ticker),
        getStockPrice(p.ticker)* p.qty)
FROM portfolio p, stockrefs r,
WHERE p.custid='bourbon' and
        r.ticker = p.ticker;
```

Figure 3-21 Accessing enterprise data via DB2 Web Services

Enterprise IMS data access via Web Services

Although IMS cannot be a Business Process Engine, we can integrate IMS into a WPS business process quite easily. Integrating IMS applications and data into the choreographed processes is possible, materialized by the IMS SOAP Gateway. IMS SOAP Gateway is a Web service solution that integrates IMS assets into an SOA. The IMS SOAP Gateway enables IMS applications to inter-operate outside of the IMS environment, through SOAP, to provide and request services enabling IMS applications to be invoked as Web Services. Therefore, existing IMS application infrastructure can be integrated in a business process and orchestrated by WebSphere Process Server (WPS).

The logical choice to run WPS is on the same platform as (or closest to) the IMS where the enterprise applications run with the availability and scalability benefits the System z platform provides. In addition, it gives the advantage of running the business process server local to MQ and IMS databases if required. The TCO advantages of running the Java workload can be realized by using the zAAP processors if IBM WPS is running on z/OS or the IFL processors if IBM WPS is running on Linux on System z.

Figure 3-22 shows two examples of exposing an IMS application via the IMS SOAP Gateway as a Web service and consumed by a Web service requester running outside of the z/OS system.

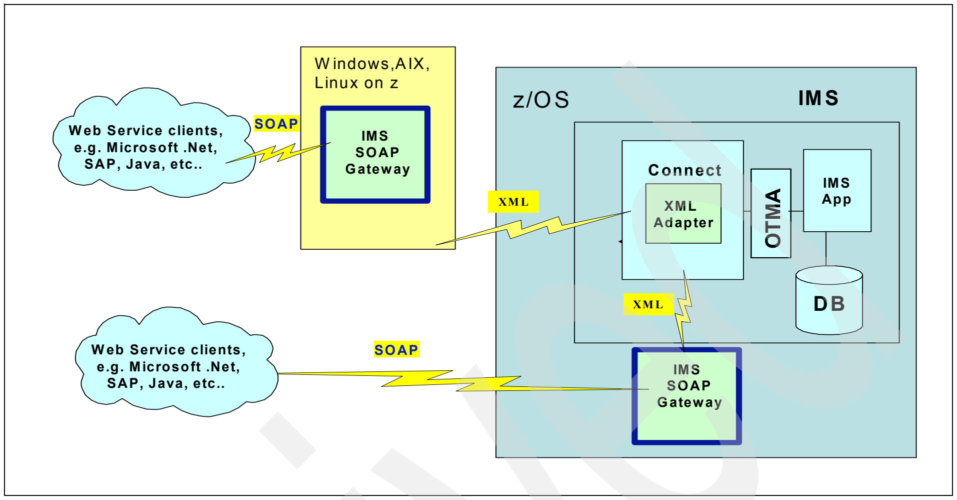

Windows,AIX, Linux on z
Web Service clients, e.g. Microsoft .Net, SAP, Java, etc..
SOAP
IMS SOAP Gateway
XML
z/OS
IMS
Connect
XML Adapter
OTMA
IMS App
DB
XML
Web Service clients, e.g. Microsoft .Net, SAP, Java, etc..
SOAP
IMS SOAP Gateway

Figure 3-22 IMS SOAP Gateway on z/OS with IMS V10

With the latest IMS v10 enhancements, Web enabling IMS data introduces a new way to view and map native IMS hierarchical data to XML documents. It aligns IMS Database (DBD) with XML Schema and allows the retrieval and storage of IMS Records as XML documents with no change to existing IMS databases. IMS Database Web Services provides a way to directly expose the IMS data as a Web service without the need to write a proxy application. The Rational tool (RAD) automatically generates a deployable WebSphere EAR file and WSDL file. The deployed Web service can now be consumed by other service requesters, a Business Process in this scenario. Figure 3-23 on page 147 shows IMS data exposed to the Business Process as a Web service.

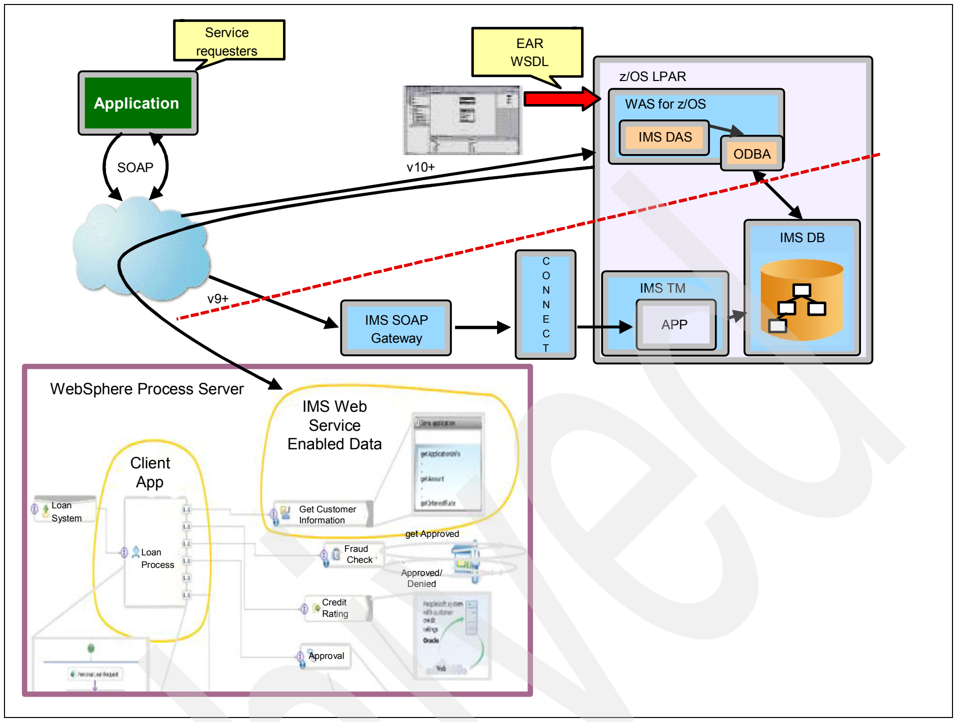


Figure 3-23 Business process with IMS data exposed as Web Services

Business process in conjunction with the Portal

Although Portal cannot be a Business Process Engine, we can integrate a WPS business process into the Portal server. In order to integrate a business process in the Portal server a business process portlet is required. A business process portlet connects a Portal application to a business process. Business process portlets are portlets that interact with the business processing code to provide task processing, process initiation, adding data (input/output messages) to business process and user interface.

This is illustrated in Figure 3-24 on page 148.

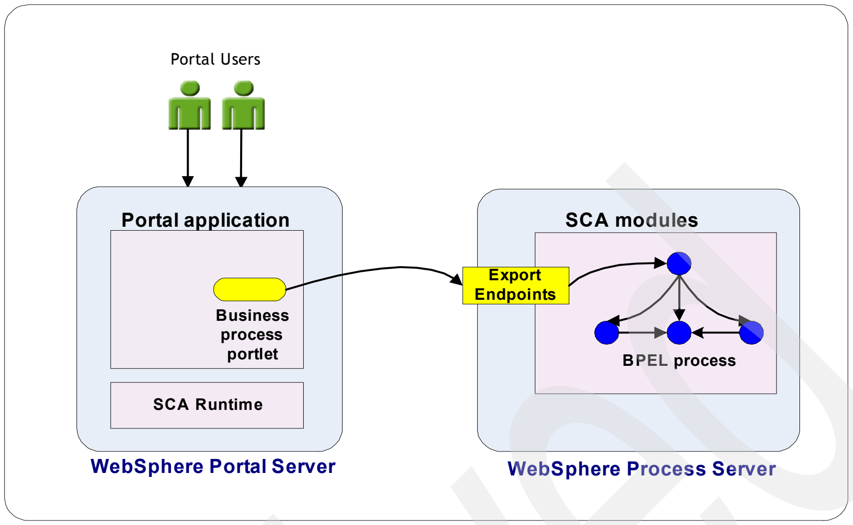


Figure 3-24 Business Process integration with Portal

There are several options to connect to the Process server:

- Using RMI/IIOP via the Business Flow Manager (BFM) or Human Task Manager (HTM) API. This requires the SCA run time on the Portal server.
- Using SOAP via the Business Flow Manager (BFM) or Human Task Manager (HTM) API. This requires the SCA runtime on the Portal server.
- Using RMI/IIOP via a locally customized facade EJB.
- Using SOAP/HTTP.
- Using JMS.

Refer to the Portal Server and Process Server documentation for more development details. We do not further discuss this in this book.

Figure 3-25 on page 149 also shows several possible options for Business Process integration with Portal.

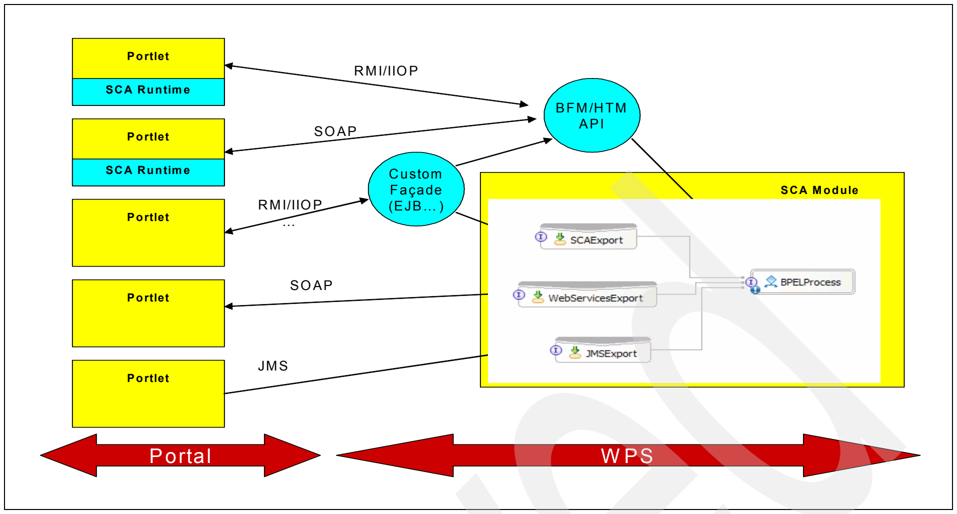


Figure 3-25 Business Process integration with Portal: Possible options

Table 3-1 presents a summary of the options and trade-offs.

Table 3-1 Portal integration options: Summary and trade-offs

Technique	**BPEL version awareness**	**Performance**	**Effort**
Portlet > BFM/HTM API > BP/HT	Yes	Good	Requires using the BO framework and the BFM/HTM client API in the portlet.
Portlet > SOAP > BFM/HTM Web Services API > BP/HT	No	Not as good as option 1	Requires using BOs and understanding of the BFM/HTM API.
Portlet > SOAP > Web Services Export > BP/HT	No	Not as good as option 1	Does not require using BOs and BFM/HTM client API.
Portlet > EJB or AJAX façade > SCA > BP/HT	Yes (only if BFM/HTM API are used)	Best	No BOs or BFM/HTM API in the portlet – but requires building the façade.
Portlet > EJB or AJAX façade > BFM/HTM API > BP/HT	Yes (only if BFM/HTM API are used)	Good	No BOs or BFM/HTM API in the portlet – but requires building the façade.
Portlet > JMS > JMS Import > BP/HT	No	Good	Only for “starter” portlets – no BO or BFM/HTM API exposure.

3.5.5 IBM WebSphere Process Server logical deployment topologies

This section describes how to deploy the IBM WebSphere Process Server components. The topologies take into consideration the hardware constraints. Business solutions can be constrained by the hardware, either in terms of the number of machines or the number of concurrent processes per machine. The final choice for a given business solution depends on the combination of component support and QoS characteristics that are needed by the business solution.

Two topologies are recommended:

- Single Cluster deployment
- Multiple Cluster deployment

Single cluster deployment

This topology is comprised of only one cluster in a cell (Figure 3-26). The cluster is the only member of the SCA System, SCA Application, and BPC buses.

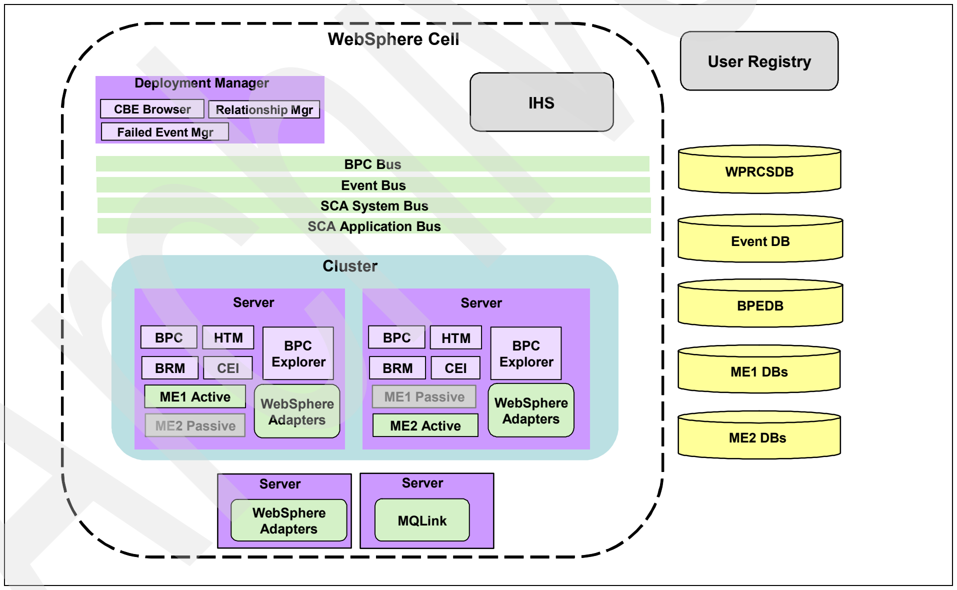


Figure 3-26 Deployment topology - Single cluster

Business applications and *Messaging Engines (ME)* reside together inside the same application server within the cluster. Therefore, there is no clear distinction between the messaging layer and the application layer.

The BPEL container is configured on this WebSphere Process Server cluster, and the cluster is a member of the SCA System, SCA Application, and BPC buses. Client applications that invoke business applications within the cluster can be either deployed into the same cell or into separate cells.

There is only one active ME on the SCA System, SCA Application, and BPC bus. The active Messaging Engine starts on one of the application servers inside the cluster. Because there is only one active ME at any given time, the MEs on other cluster members are in joined state. The *High Availability Manager (HAM)* decides which server runs the active ME. On error of the active ME, HAM starts an ME on a joined server. Persistent messages in the data store are now accessible again through the started ME.

More MEs can be added to a bus member with *Partitioned Queues*. Partitioned Queues imply every logical queue destination has n queue points, each of n partitions perhaps holding an nth of the messages. It provides scalability if the bus member is a cluster. However, the message ordering is not preserved.

Advantages of a single cluster include the following:

- Easier to install, configure and manage (fewer servers and server components).
- Smaller memory footprint on z/OS provides for greater throughput if it was deployed on a z/OS environment.
- Shorter recovery and restart times because there are fewer servers to recover and restart.
- Better transaction rate for the entire WPS workload.
- Cost per transaction is lower in the single cluster deployment.

Multiple clusters deployment

This topology is comprised of one Messaging Engine cluster, one Support cluster, and one or more Application clusters. In Figure 3-27 on page 152 the Messaging Engine cluster is the only member of the SCA System, SCA Application, BPC, and CEI bus. The business applications and the Messaging Engines reside in different application servers within the cluster. Therefore, there is a clear distinction between the messaging and the application layer. There is only one active Messaging Engine on the SCA System, SCA Application, BPC, and CEI bus. This Messaging Engine is located within the Messaging Engine cluster.

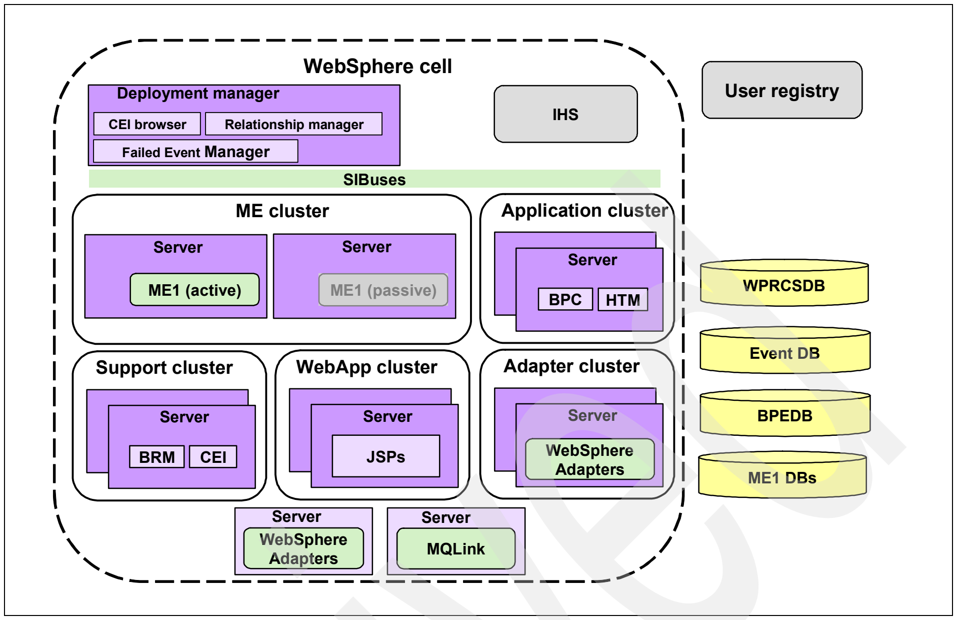


Figure 3-27 Deployment topology - multiple clusters

In this topology, applications and the ME are split into dedicated clusters, that is, application clusters and ME cluster. An application interacts with another application asynchronously within the same server. All applications within the same application cluster write message to the ME located on the server where the active ME is running. Both the ME and the applications are highly available. Applications are also highly scalable, but the ME is limited in scalability.

Adding more MEs to a bus member with Partitioned Queues will make the ME become scalable but an unprocessed message situation will occur. A combined application and ME cluster can resolve this issue.

Advantages of a remote ME cluster include the following:

- More even distribution of asynchronous and synchronous workload in the application cluster.
- WPS applications are isolated from the ME infrastructure components.

3.5.6 IBM WebSphere Process Server deployment configurations

At the time of writing, there are two IBM WebSphere Process Server versions that support IBM System z:

- IBM WebSphere Process Server v6.1 for z/OS
- IBM WebSphere Process Server v6.1 for Linux on System z

Linux on System z deployment

Figure 3-28 shows the IBM WebSphere Process Server deployed on a Linux on System z environment.

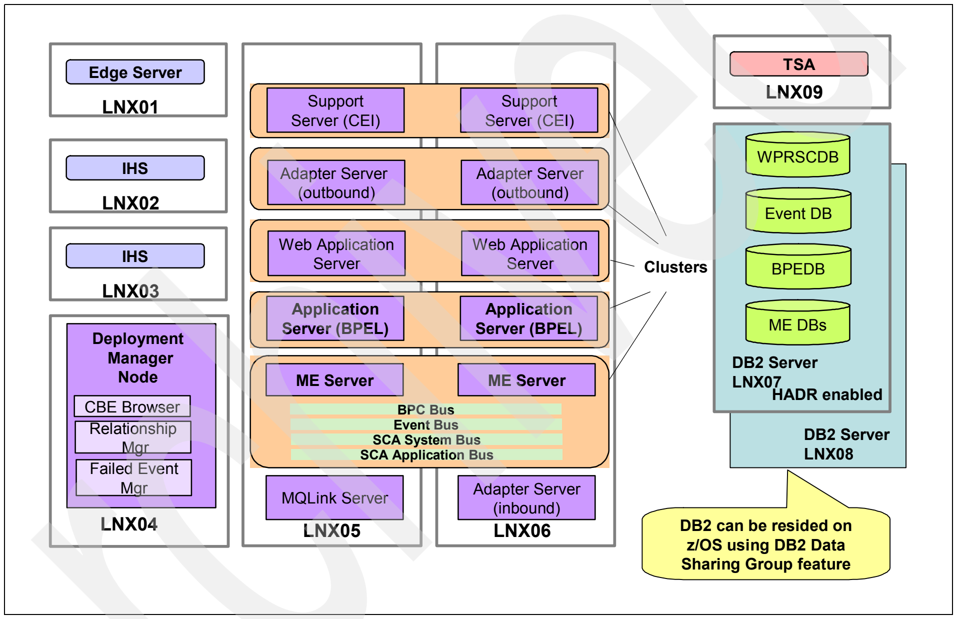


Figure 3-28 WPS - Linux on System z deployment

Linux on System z deployment is very similar to other distributed platforms, such as UNIX and Windows. The only difference is that all the Linux servers are virtual guests running under z/VM on the IBM System z hardware. In this configuration, there are 9 Linux on System z servers installed and they are all running under z/VM as second level guests. The WebSphere Process Server is configured with multiple clusters under a WebSphere Application Server ND cell that spans across two Linux on System z servers. One Linux on System z server will be used as the Deployment Manager node with 3 WPS applications as shown in the diagram.

There is also a pair of IBM HTTP servers with the WebSphere Application Server Plug-in configured, and inbound HTTP traffic is load balanced by an IBM Edge Server in front of it. In a production environment, the Edge Server and the IBM HTTP servers will be residing on the enterprise DMZ, which is facing the customers on the Internet. A second Edge Server (Edge Server clustering) for failover can be configured as well, but this is not shown in the diagram.

There are two Linux on System z servers for DB2 LUW (Linux, UNIX, WIndows) with HADR configured for data replication and recovery. The *Tivoli System Automation (TSA)* server will be used to automate the DB2 server's fail over and recovery.

In this configuration, instead of using the DB2 LUW, the WebSphere Process Server database can be on the z/OS LPARs with the *DB2 Data Sharing Group (DSG)* enabled for load balancing and failover capability. Hipersockets can be used in this scenario for the network traffic between the Linux on System z servers and the z/OS DB2 LPARs.

z/OS deployment

In this configuration, IBM WebSphere Process Servers are configured on two z/OS LPARs. The deployment topologies described in "Single cluster deployment" on page 150 and "Multiple clusters deployment" on page 151 are discussed.

Figure 3-29 on page 155 shows the WebSphere Process Servers configured in a single cluster with one active Message Engine (ME) running on one LPAR. There is only one active ME on the SCA System, SCA Application, and BPC bus. The active Messaging Engine starts on one of the Application Servers inside the cluster. The MEs on other cluster members are in a joined state. The High Availability Manager (HAM) decides which server runs the active ME. On error of the active ME, HAM starts the ME on a joined server. Persistent messages in the data store are now accessible again through the started ME. Only the local process application can consume the messages from the active ME.

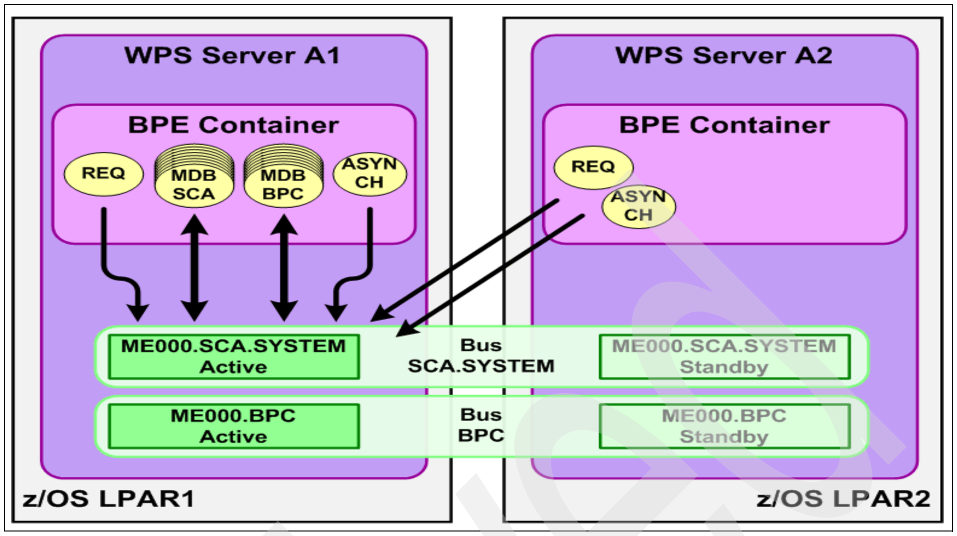


Figure 3-29 Simple cluster in z/OS

For the WebSphere Process Server database and the ME database, a DB2 Type 2 connection and a DB2 Data Sharing Group (DSG) are used.

There are benefits and limitations in this scenario:

- The benefits are:
  - High availability of the Message Engine and the WPS components
  - Efficient DB2 Type 2 connection and DB2 data sharing for database access
- The limitations are:
  - No workload sharing due to process applications only running on the server where the Message Engine is active
  - No scalability for Message Engine due to only one active Message Engine instance

We now discuss a scenario with multiple clusters.

Figure 3-30 on page 156 shows one Messaging Engine cluster and one application cluster. There is only one active ME on the SCA System, SCA Application, and BPC bus. The active Messaging Engine starts on one of the application servers inside the cluster. Similar to the single cluster configuration, the High Availability Manager (HAM) decides which server runs the active ME. On error of the active ME, HAM starts the ME on a joined server. Persistent messages in the data store are now accessible again through the started ME. In this scenario, the process application running on the other LPAR can consume the messages from the active ME remotely.

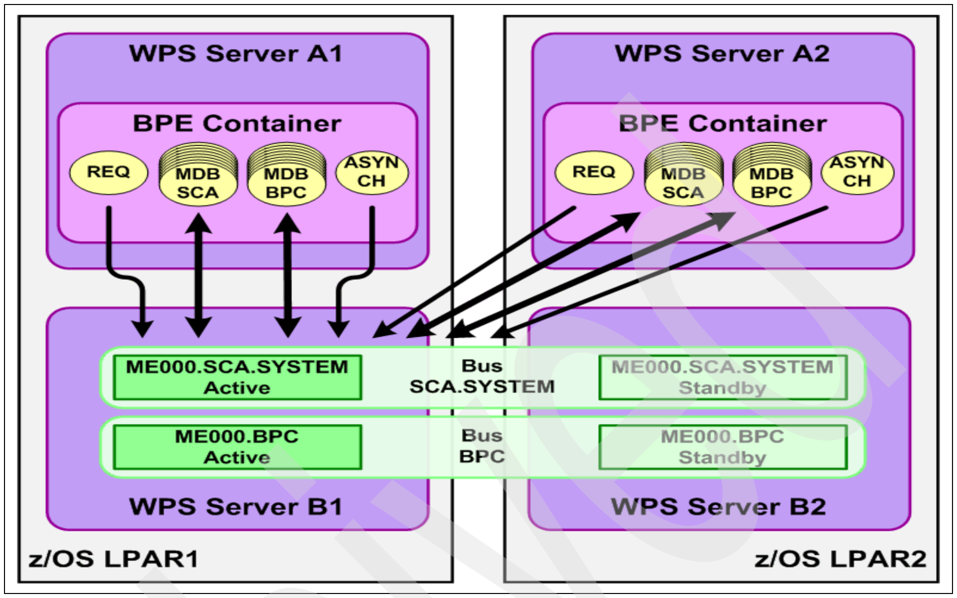


Figure 3-30 Two clusters in z/OS

As in the single cluster configuration, DB2 Type 2 connections and DB2 data sharing is used for accessing the WebSphere Process Server database and the ME database.

However, in order to avoid unnecessary overhead on z/OS, only two clusters are needed when applying the multiple clusters topology to z/OS. We recommend that only the Message Engine cluster is created. Keep all the other support servers in another cluster.

In this scenario, there are benefits and limitations:

- The benefits include:
  - High availability of Message Engine and of WPS components.
  - Efficient DB2 Type 2 connection and DB2 data sharing for database access.
  - Applications can run on both z/OS LPARs regardless of where the Message Engine is active. (However, it is not recommended for z/OS due to the additional CPU overhead on z/OS.)

- Limitations include:
  - No scalability for Message Engine due to only one active Message Engine instance.
  - Greater memory footprint on z/OS.
  - High CPU overhead if all the WPS components are in separated clusters.

Figure 3-31 shows a configuration similar to the single cluster configuration except that the queues are partitioned. Additional Message Engines can be added to the bus member with Partitioned Queues. Partitioned Queues imply every logical queue destination has n queue points, each of n partitions perhaps holding an nth of the messages. It provides scalability if the bus member is a cluster.

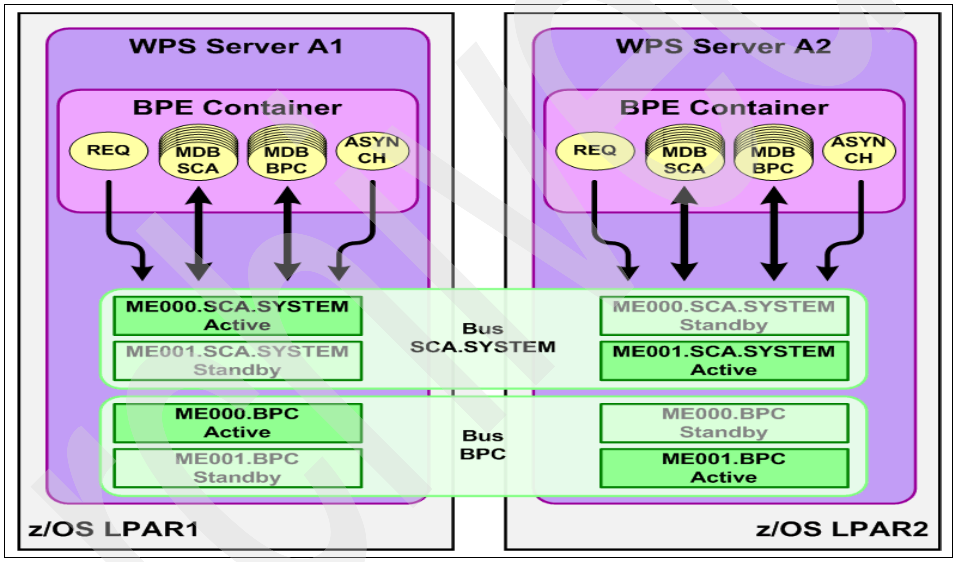


Figure 3-31 Single cluster with partitioned queues in z/OS

As in the other configurations, DB2 Type 2 connections and DB2 data sharing are used for accessing the WebSphere Process Server database and the ME database.

The benefits in this scenario are the following:

- High availability of Message Engine and of WPS components.
- Efficient DB2 Type 2 connections and DB2 data sharing for database access.
- Applications can run on both z/OS LPARs with workload being shared.
- High scalability of Message Engine and WPS components.

Figure 3-32 shows a single cluster setup. Instead of using the WebSphere SIBus and the Message Engine, it leverages the fast and highly resilient z/OS messaging infrastructure based on *WebSphere MQ (WMQ)* for z/OS, including the use of *MQ Shared Queues*, which is only available for z/OS. It ensures high availability and workload balancing. Similar to the previously discussed configurations, DB2 Type 2 connections and DB2 data sharing are used, but only for accessing the WebSphere Process Server database.

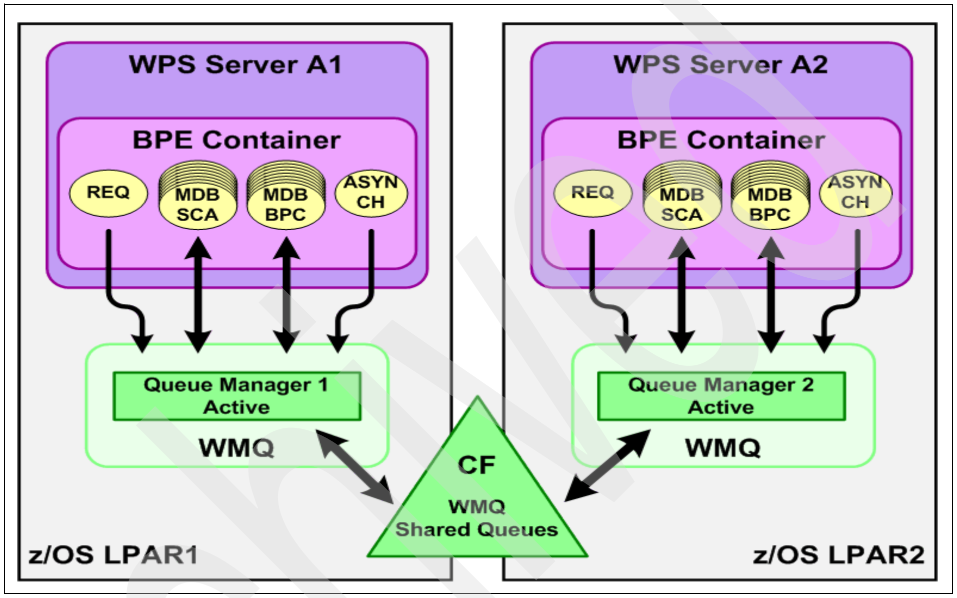


Figure 3-32 Single cluster and WebSphere MQ shared queues

In this scenario, there are benefits and limitations:

- The benefits are:
  - High availability of WPS components.
  - Efficient DB2 Type 2 connections and DB2 data sharing for database access.
  - Applications can run on both z/OS LPARs with workload being shared.
  - Fast and highly resilient messaging infrastructure (WMQ) is used.
- Limitations include:
  - Requires an additional amount of configuration overhead on WMQ setup for Shared Queues.

3.6 Dynamic BPM solution for composite business applications

In “Business process services solutions” on page 129, we discussed the Business Process services solutions that focus on service orchestration. In this section, we discuss a solution that will enable enterprises to dynamically compose and decompose applications according to business needs. The dynamic BPM solution focuses on dynamic processes, the next step in delivering agile business solutions. A dynamic BPM solution can quickly combine existing business services to offer new services. This dynamic solution is sometimes referred to as *service-oriented business applications (SOBA)* or solutions, and it extends the SOA life cycle to business services. It also provides the ability to build business processes that are more responsive to end-user needs, easier to manage, and quicker to change. The solution will create and run flexible composite business applications to support the core business operations. There are different levels of abstraction within BPM and SOA, leading to increased levels of business agility.

At the very basic level are Web services and components. Web services help to simplify integration and are typically where companies start out with their SOA initiatives. At this level, BPM automates processes by basically orchestrating services and connecting them together into business processes. This is what we have discussed in the previous Business Process services solutions section.

As we move to process choreography capabilities, each Web service involved in the choreography knows exactly when to execute its operations and with whom to interact.

But to get to the ultimate level in business flexibility, we want to get to dynamic composition, where we use business context to dynamically determine behavior. Here is where we define business services that abstract this information and become the building blocks of composite business applications. The IBM product *WebSphere Business Services Fabric (WBSF)* extends IBM’s ability to deliver BPM enabled by SOA. IBM WBSF is a software platform to enable a new class of service-oriented business solutions. It provides modeling, assembly, deployment, management, and governance of business services and includes optional “Industry Content Packs” that contain pre-built SOA assets that accelerate development of industry-specific service-oriented applications. WBSF provides dynamic assembly and delivery of services based on business context. It delivers flexibility and responsiveness across multiple constituents, channels, and service levels based on reusable, business-level building blocks. One of the WBSF benefits is that it allows you to continue using existing IT investments wherever possible, leveraging existing I/T investments in new flexible ways for new advantage in the market. In other words, do not rip and replace.

3.6.1 WebSphere Business Services Fabric runtime architecture

The WebSphere Business Services Fabric runtime architecture consists of the following components, which provide comprehensive dynamic process capabilities:

- *Business Services Composition Studio* is a plug-in to WebSphere Integration Developer (WID).
- *Business Services Dynamic Assembler* is an SCA module that runs within WebSphere Process Server and leverages WebSphere ESB.
- *Business Services Repository* federates with WSRR.

Figure 3-33 shows the runtime architecture of WBSF.

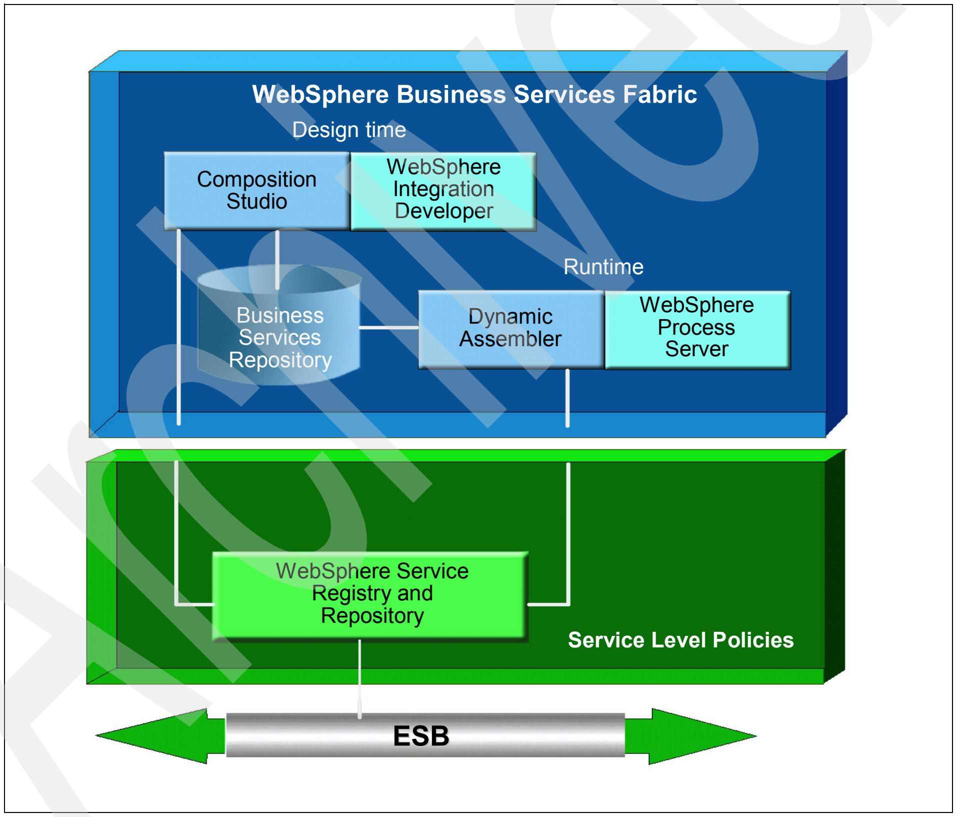

WebSphere Business Services Fabric
Design time
Composition Studio
WebSphere Integration Developer
Runtime
Business Services Repository
Dynamic Assembler
WebSphere Process Server
WebSphere Service Registry and Repository
Service Level Policies
ESB

Figure 3-33 WBSF runtime architecture

The *Business Services Composition Studio* is a "standalone" Eclipse plug-in used to describe services and composite business services by means of the attributes, and to describe relationships among services, subscribers, and policies. The

Composition Studio enables domain-experienced software architects to model, create, publish, and manage industry-specific service metadata models and policies around data, processes, resources, policies, and domains. It provides the capability to:

- Compose and publish composite business services metadata
- Create declarative policies for endpoint selection and service personalization based on content, context, and contract
- Simulate service behavior based on different usage scenarios
- Auto-discover Web services from registries
- Validate meta models to ensure accuracy and correctness

The *Business Services Dynamic Assembler* is a feature of WBSF that uses metadata and policies to mediate between business service requesters and providers at run time. This feature complements the existing service composition capabilities in the WebSphere Enterprise Service Bus by adding a declarative service mediation engine that uses the business semantics and policies captured in the IBM Business services meta-model and stored in the WebSphere Business Services Repository to dynamically mediate between service requesters and service providers. The Business Services Dynamic Assembler performs dynamic service selection and personalization based on context, contract, and content. It can be integrated with the ESB and BPEL runtime environments. It supports HTTP, EJB, JMS, SOAP, and non SOAP messages. It also can be integrated with existing System Management platforms such as Tivoli Enterprise Console® and ITCAM for SOA.

The *Business Services Repository* is a standards-based, enterprise SOA metadata repository for business service descriptions, ontologies, subscribers, and policies. It enables the rich description, discovery, and federation of data across universal description, discovery, and integration (UDDI) registries, repositories and Lightweight Directory Access Protocol (LDAP) systems. It can federate metadata from other repositories such as LDAP systems, WSRR, and UDDI registries with full support for versioning. The Business Services Repository is capable of segmenting the CBS metadata by Namespace and control its visibility, automating rule-based validation on client and server, and is able to do fine grained rollbacks. It provides conflict detection during collaborative development. The OWL/RDF based meta-model allows for capturing semantics and provides powerful search, dependency, and impact analysis.

3.6.2 WebSphere Business Services Fabric base packs

WebSphere Business Services Fabric consists of a "Tool Pack" and a "Foundation Pack." The Tool Pack is the design time environment consisting of Business Services Composition Studio as an Eclipse plug-in into WID. The

Foundation Pack is the run time and manage time components, which are deployed into WPS. They are packaged as a set of J2EE components. The WebSphere Dynamic Process Edition also includes the Business Modeler and the Business Monitor.

Figure 3-34 shows the components of the WBSF base packs.

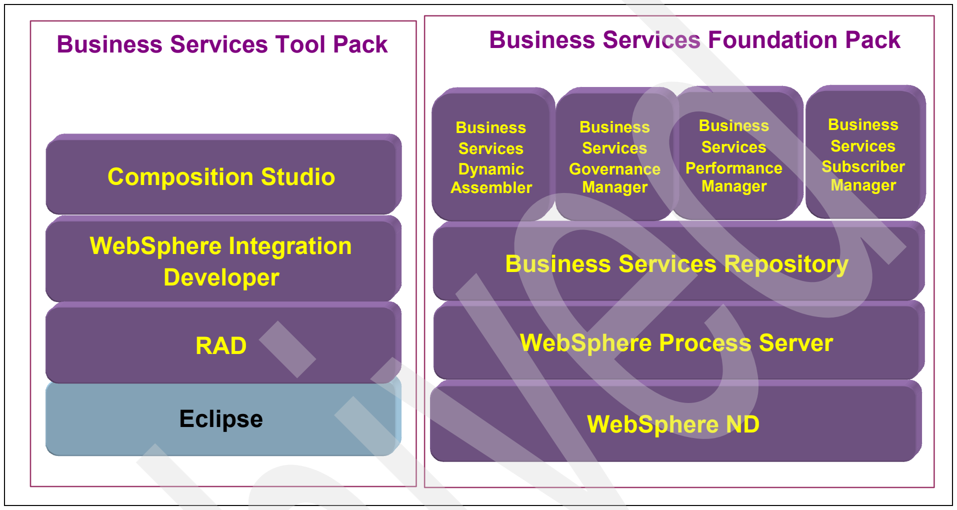

Business Services Tool Pack
Composition Studio
WebSphere Integration Developer
RAD
Eclipse
Business Services Foundation Pack
Business Services Dynamic Assembler
Business Services Governance Manager
Business Services Performance Manager
Business Services Subscriber Manager
Business Services Repository
WebSphere Process Server
WebSphere ND

Figure 3-34 WBSF base packs

Business Services Composition Studio and WebSphere Integration Developer are shipped with the Tool Pack and have already been discussed in "WebSphere Business Services Fabric runtime architecture" on page 160. The Business Services Dynamic Assembler and the Business Services Repository are shipped with the Foundation Pack and have also been discussed previously.

Business Services Governance Manager provides governance of Business services metadata throughout its life cycle. It manages life cycle changes to CBS, controls metadata promotion between environments, and validates meta models to ensure accuracy and correctness before publishing to the Business Services Repository. It provides open APIs for integration with enterprise change management systems such as Rational ClearCase®, defines policies for metadata visibility during collaborative development, and defines and manages namespaces to segment metadata in the Business Services Repository. It notifies users when changes are made and configures federation settings for WSRR and LDAP integration.

Business Services Subscriber Manager controls and automates entitlement of business services for subscribers. It enables the creation, control, and management of service portfolios for subscribers across a business ecosystem, and integrates with leading security and identity management products. It manages business service portfolios for subscribers; federates subscriber information from LDAP systems for users, roles, and groups; represents complex organization hierarchies; captures metadata relating to subscribers; and supports self-service and assisted enrollment models.

Business Service Performance Manager provides visibility and monitoring of service-oriented processes and applications. This module includes multi-perspective views and enables drill down analysis of business level events and exceptions. It provides Business service performance and SLA reporting, subscriber and role-based drill down of Business service performance, and billing and metering of Composite Business services at various levels. It also can be integrated with existing System Management platforms such as Tivoli Enterprise Console (TEC) and ITCAM for SOA.

3.6.3 WebSphere Business Services Fabric solution packaging

In addition to the Tool Pack and Foundation Pack, WebSphere Business Services Fabric offers optional Industry Content Packs that contain pre-built SOA assets and templates that accelerate deployment of industry-specific composite business applications.

Industry Content Packs (ICP) are collections of pre-built reference business services templates. They are optimized for use with a specific industry's ecosystem (customers, partners, employees, IT systems, and so forth) and are based on industry and technical standards. They are extensible and configurable to support unique client needs.

Industry Content Packs are IBM products that are maintained, updated, and released. However, Business services can be deployed on the Fabric without an Industry Content Pack. Currently, IBM offers four ICPs:

- IBM Healthcare Payer Content Pack
- IBM Insurance P&C Content Pack
- IBM Banking Payments Content Pack
- IBM Telecom Operations Content Pack

Figure 3-35 on page 164 shows the role of the ICPs.

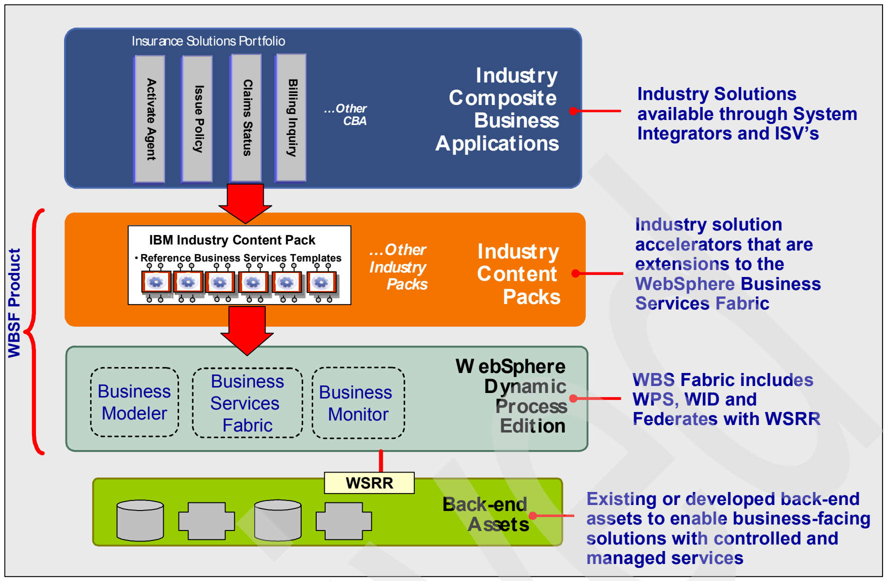

Insurance Solutions Portfolio
Activate Agent
Issue Policy
Claims Status
Billing Inquiry
...Other CBA
Industry Composite Business Applications
Industry Solutions available through System Integrators and ISV's
IBM Industry Content Pack
• Reference Business Services Templates
...Other Industry Packs
Industry Content Packs
Industry solution accelerators that are extensions to the WebSphere Business Services Fabric
WBSF Product
Business Modeler
Business Services Fabric
Business Monitor
WebSphere Dynamic Process Edition
WBS Fabric includes WPS, WID and Federates with WSRR
WSRR
Back-end Assets
Existing or developed back-end assets to enable business-facing solutions with controlled and managed services

Figure 3-35 WBSF Industry Content Packs (ICP)

4

Business Application services

In this chapter we provide scenarios and solutions to consider when developing an architecture necessary to provide Business Application services on System z. This chapter also discusses various scenarios to consider when both new and existing System z based business application functions are enabled to participate as services in a service-oriented architecture (SOA).

We discuss how the anatomy of existing functionality might influence the granularity level of services and to what extent modernization or rejuvenation of such functionality might assist in mitigating this problem.

We also cover aspects of developing new functionality that will participate in an SOA with specific focus on the suitability of a System z platform to host such functionality. Here such items as proximity of a service-oriented business application (SOBA) to an enterprise datastore and an Enterprise Service Bus (ESB) are considered.

This chapter is structured as follows:

- In 4.1, "Definitions, scope, and functions" on page 166 we describe the scope and functions of the Business Application services layer as part of the IBM SOA Reference Architecture.

- Once you have decided to implement Business Application services on System z there will be some important decisions to make. We discuss these decisions in 4.2, “Key architectural decisions with respect to Business Application services on System z” on page 168.
- In 4.3, “Common application patterns” on page 175 we discuss a number of common application patterns on System z (mostly z/OS) and how they impact SOA.
- In 4.4, “Platform and runtime container choice criteria” on page 184 we discuss the most important criteria with regards to the platform, O/S and container of the future Business Application services.
- In 4.5, “Solutions” on page 189 we give an overview of technology to enable running Business Application services on System z.
- In 4.6, “Financial implications” on page 214 we mention a few considerations from a financial perspective.
- In 4.7, “Challenges and issues” on page 215 we summarize the most important issues and challenges in building your Business Application services.

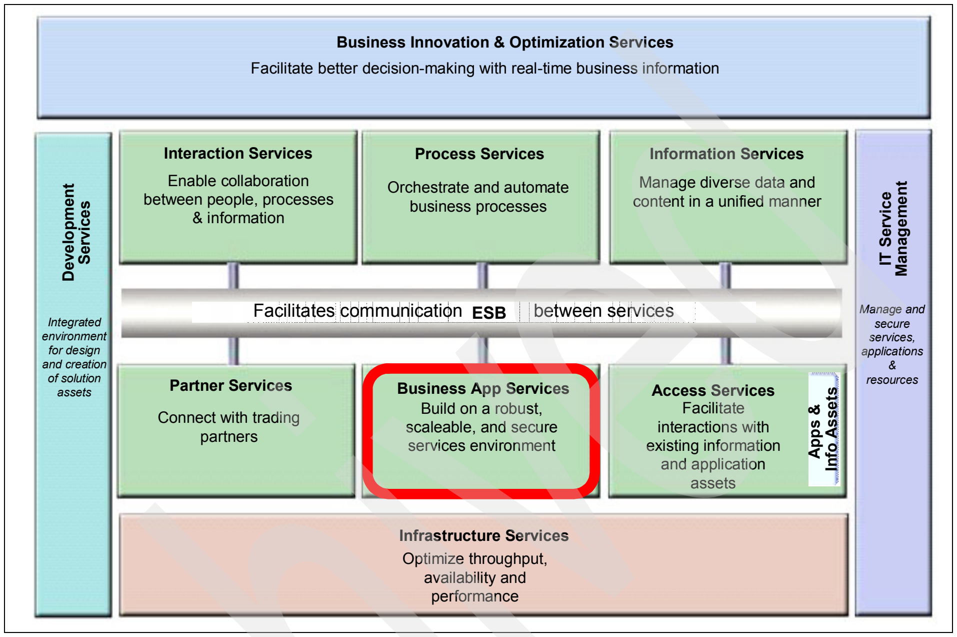
4.1 Definitions, scope, and functions

4.1.1 Definition

Business Application services implement core business logic. Loosely coupled business components can be combined using process orchestration to provide a complete business process that can participate in a service-oriented business application (SOBA). Business Application services can be constructed either by refactoring existing applications or by the development of new functionality. Existing business applications on System z are typically implemented in third generation languages (3-GL) such as COBOL or PL1 running under a TP monitor or container such as IMS or CICS. Data is usually hosted within a DBMS such as in DB2 or IMS, or files such as VSAM or QSAM. However, J2EE applications implemented in WebSphere Application Server on z/OS or Linux on System z are becoming more common. Here the DBMS of choice is usually DB2, but IMS databases or VSAM files can be accessed.

4.1.2 Scope

Business Application services provide access to the realization of a business application as implemented on System z. Business Application services implement core business functionality. These services interact with other SOA components such as business process services and Enterprise Service Bus (ESB) services, among others. Discussions about these components are covered in their respective chapters within this book. Figure 4-1 locates Business Application services within the SOA reference architecture.



Business Innovation & Optimization Services
Facilitate better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enable collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manage diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
Facilitates communication
ESB
between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitate interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimize throughput, availability and performance

Figure 4-1 Business Application services in the IBM SOA reference architecture

4.1.3 Functions of Business Application services

The function of business application services is to provide the core processing components as required within an SOA implementation. They can be indirectly exposed to consumers via an ESB or they can be directly exposed. Business Application services can also form components within a larger orchestration or service composition provided via *Business Process Management (BPM)*.

Business Application services should ideally be loosely coupled and stateless. The ability to seamlessly form part of a complete service-oriented architecture as well as participating within a distributed atomic transaction requires extensive communication capabilities as well as offering these services to consumers via an industry standard protocol such as SOAP.

Business Application services are a key components within SOA. The overall Quality of Service (QoS) required and expected of such services is extremely high. Table 4-1 details some of the core requirements that might form part of business application service non-functional requirements (NFRs).

Table 4-1 Examples of Business Application services NFRs

Category	Capabilities	Description
Communication	► Asyncronous ► Non proprietory	Delivery of messages to core logic in a loosely-coupled manner using industry standard protocols. This will enable better reuse of services.
Integration	SOAP based	Supports integration of heterogeneous environments and therefore ease of service composition. Supports SOA principles, separating application code from specific service protocols and implementations.
Service availability	► Failover ► Workload balancing	Ability to meet agreed SLAs and prioritize workloads. The runtime container of the Business Application services will need to be able to meet these SLAs, enabled by the O/S and the hardware.
Service scalability	Vertical and horizontal scalability provided by runtime container, O/S, and the hardware.	The ability to dynamically increase system resources and dynamically replicate Business Application services containers and servers is required.
Security	Authentication and authorization	Business Application services run time might need to facilitate authorization and authentication to the service. However, this function might be delegated to an ESB or dedicated security solution.
Management	Administration capability	A point of control over service availability.

4.2 Key architectural decisions with respect to Business Application services on System z

The most important discussion regarding Business Application services on System z concerns reusing existing assets. Most SOA implementations on

System z will be in a position to benefit from extensive reuse of existing legacy functions. There is currently sufficient technology and methodology available to do this.

Whether Business Application services are composed of newly developed functionality or existing functionality that requires modernization and integration might influence your approach to implementing SOA. These differing approaches in turn influence platform choice and tooling. To understand the process of establishing Business Application services you have to consider whether infrastructure and non-functional requirements should be applied to a service at the service level of granularity or to an application-wide view.

In this section we demonstrate how architectural decisions influence the resulting topography of the SOA landscape. These decisions are:

- What will be the *SOA approach*?

  Business Application services must be considered within the context of the overall SOA. Choices in the ESB, business process services, or information services layers of the SOA can impact decisions about Business Application services and vice versa. Implementing SOA in an existing application environment, which is usually the case on System z, you must choose between following a top-down, bottom-up, or middle-out approach. Also, an approach or method for service identification must be chosen. A top-down approach is driven from the business requirements and by business analysts, whereas bottom-up is based on analysis of existing applications and usually from an IT perspective and initiative. Middle-out is a combination of the two processes cemented with a final gap analysis.

  Refer to “A top-down versus bottom-up approach” on page 170 for a more detailed discussion.

- What is the approach to integrating existing assets in an SOA?

  Options are *refacing*, *rejuvenating,* or *replacing* existing applications. Some companies also consider *replatforming* as an option.

  – *Refacing* implies modernizing the user interface by replacing the green screen UI with a Web browser UI.

  – *Rejuvenating* is a process in which an existing application is “tuned” to be integrated into an SOA. The key objectives are to achieve an optimal service granularity and clear demarcation of presentation, business, data access, mediation, and workflow logic

  – *Replacing* an existing application implies a full re-write of an application into reusable services and eventually deploying it on another platform. Another option is to replace an existing legacy application with a purchased ISV application with SOA integration capabilities.

Refer to “Rejuvenate, reface, or replace?” on page 173 for a more detailed discussion.

- What *programming model* will be used for new Business Application services?

  There are only a few real choices nowadays on System z. They are J2EE; any supported 3-GL, such as COBOL and PL/I; or a 4-GL, such as IBM’s Enterprise Generation Language (EGL). Combinations are possible.

  Refer to “Programming model” on page 174 for a more detailed discussion.

- As discussed in other chapters, the *location* of the Business Application services needs to be decided. Different Business Application services can be placed in different locations, mostly determined by the location of the data and other services that the business application service integrates with.

  There are basically two philosophies: one embraces the view that distribution of Business Application services and data does not really matter, and one advocates centralization of Business Application services and data as much as possible. Using z/OS on System z clearly fits in the latter philosophy. The vision behind this philosophy is better Quality of Services, higher ability to meet SLAs, and a better handle on cost.

4.2.1 A top-down versus bottom-up approach

In this section we discuss the differences between top-down and bottom-up SOA approaches.

Top-down

A *top-down* approach is one that is driven from a set of business requirements. Services are identified from these requirements. We believe the top-down approach actually has two different patterns.

The first pattern is one of a business-led SOA initiative. Business units across an organization have embraced the SOA concept and its benefits and are typically funding and driving this initiative. Adoption of this pattern is usually a good indicator that an enterprise-wide view of services is being taken, with a good probability of reuse for these services.

A second pattern is one where a specific project is being driven using a top-down approach. This is similar to a silo approach. Being project-driven, this pattern has more likelihood of delivering services that suffer from tunnel vision. The service might satisfy a specific business requirement, but it is not reusable across the enterprise. In the second pattern a wide-eyed approach needs to be ensured by continuous reviews by SOA analysts. Here good governance is essential.

Bottom-up

A *bottom-up* approach is driven from the analysis and decomposition of existing functionality. This approach is often driven from within IT and might even be funded internally within IT. The process of mining functionality and then refactoring of these fragments into services can provide a fast turnaround. However, because the process is driven from within IT, review procedures should be in place to ensure services are able to satisfy real business processes. Of real benefit, however, is that because there are no specific business requirements driving the process, the likelihood of an SOA analyst producing an enterprise-wide view for a service is more likely.

What tends to be most successful is when a bottom-up approach is first adopted by IT. Here the initial services are often self evident. This initial strategy can allow for organizational change and SOA adoption by business units. A top-down approach can now also be adopted in parallel, resulting in a form of “meet in the middle” or middle-out approach between IT and business units.

Middle-out

The *middle-out* approach is, as the name implies, a hybrid of the two approaches just discussed. Running in parallel with the bottom-up approach, top-down business decomposition is undertaken, yielding use cases and functional requirements. At the same time, the bottom-up approach yields a set of components that already exist in applications. This is followed by a gap analysis to determine which business functions cannot be satisfied with the existing components and therefore require new development. This approach is valuable in that it enforces engagement between business and IT. This has value in ensuring an enterprise-wide view instead of a project-specific view is adopted in service realization. It also helps to ensure that services are not realized from a bottom-up approach that have little business benefit or probability of reuse. This middle-out approach, however, requires a corporate structure and support that often is slow to achieve. Nevertheless, it should be a primary goal in an SOA roadmap.

While the merits of a top-down versus bottom-up approach to SOA require considerations regardless of platform, there are typically various architectural aspects specific to the System z platform. The SOA approach chosen will be heavily influenced by whether application functionality exists or is still to be developed. We look at these scenarios in the following sections.

SOA approach and existing assets

Where business application functionality already exists, a certain amount of flexibility is required in deciding on a top-down, bottom-up, or middle-out approach.

Top-down approach

In the top-down scenario a new business requirement can be satisfied from existing logic. This assumes the existing business functionality is well understood and documented. If this is not the case an analysis process must be undertaken to understand the current business functionality and any encapsulated process choreography.

The likelihood of significant refactoring in existing assets is high with the top-down approach because the services are being redefined from a pure business perspective. Many existing assets are “polluted,” meaning they serve functionality that could be part of multiple different services or implement irrelevant or redundant functionality. One of the biggest challenges in reusing assets in a top-down approach is to align the existing assets to the business services being defined.

Bottom-up approach

As stated earlier, a bottom-up approach is driven by IT, but might be sponsored and supported by the business. A bottom-up approach seems very attractive in a landscape with many clearly reusable assets. A strong consideration for service enablement of existing functionality is the ability to achieve the correct level of service granularity. This is influenced by such things as monolithic programs and 3270 screen flows. Therefore, the decision to integrate an existing function or modernize it might influence your ability to achieve an optimal level of service granularity from the resulting components. Later in this chapter we discuss techniques and tooling to enable application modernization or refactoring.

Middle-out approach

With the middle-out approach, impact on reusing existing assets is medium because this approach is a blend of the other two approaches. Refactoring of existing assets is still very likely with this approach, but there will also be assets that can be service enabled “as is.”

New services

Where a new business application service or component is to be developed, the logical approach is top-down. As mentioned previously, the adoption of SOA within the business unit is paramount to achieving the enterprise-wide view. If this is not the case the granularity level of a service, its suitability for reuse, or its inclusion within other business application compositions might be jeopardized.

Some key architectural decisions when determining the optimal platform and infrastructure for a new service are:

- Proximity of the new service to other existing services it inter-operates with
- Proximity of the new service to the enterprise data store

- Atomic transactionality
- Platform and language choices available
- Business application service containers available and their characteristics and qualities

This topic is discussed in more detail in "Platform and runtime container choice criteria" on page 184.

4.2.2 Rejuvenate, reface, or replace?

Rejuvenation or modernization of an application deals with the process of taking existing, possibly monolithic applications and refactoring them into suitable components that can participate in a business application service. The complexity of this process depends largely on the nature of the application. If presentation logic is encapsulated within an application it might present a more complex scenario in order to achieve modernization. Presentation logic within a typical System z based application is either 3270-based or a proprietary format. For the sake of clarity, we refer to an application whose request and response format does not contain any presentation attributes as "datastream only." A datastream refers to a simple byte array containing no embedded metadata or presentation attributes. Other terms for this are bitstream, fixed width, and so forth. Such applications might be referred to as "presentation agnostic." CICS-based applications displaying such characteristics are often referred to as "COMMAREA only" applications.

Whether the application is monolithic or modular influences the techniques and tooling you will require to compose business components that can participate in a service.

If the application you wish to service enable is both modular and presentation agnostic, you are well positioned to compose services from these components. Applications without these attributes might require modernization. While there is often resistance to such modernization, we believe the benefits it offers to SOA enablement far outweigh the drawbacks. We do, however, accept that this approach might not be available in some cases, and therefore would require some form of integration process. An application that will continue to contain 3270 attributes will require specific message transformation and interpretation (often referred to as screen scraping). Applications that remain with presentation logic might still, however, be useful components in service enablement.

Monolithic applications present a more complex problem. While it might not be obvious, these applications are often composed of multiple components exposed only as one complete business process due to process choreography embedded within logic. This might affect your ability to achieve an optimal granularity level for services. A sub-optimal level of granularity diminishes the ability to form business processes from the composition of fine-grained Business Application services. This ability to compose complete processes using service composition is fundamental to achieving the flexibility and responsiveness offered by SOA and so eagerly sought by business.

Besides embedded presentation logic and being monolithic, another complexity factor in application modernization is the presence of clear demarcation between the different types of logic in the application or component. Ideally, embedded workflow and mediation logic are moved to the business process services and ESB services layers respectively. Data access logic might need to be converted in data access services or information services.

The different approaches are dealt with in more detail in the relevant sections later in this chapter.

4.2.3 Programming model

On System z, both in z/OS and Linux on System z, there is choice in programming models. Service components can be developed using the following programming models. Not all programming models are possible in certain runtime containers, therefore the programming model choice and the runtime container choice for the Business Application services go hand in hand. The programming models are:

- Java 2 Enterprise Edition (J2EE) programming model

  This programming model can be used when running the Business Application services in J2EE-compliant application servers. Practically all J2EE application servers provide extensive Web Services capabilities.

- COBOL

  COBOL is still a widely used language and has undergone significant modernization. One of the latest enhancements in IBM's Enterprise COBOL is support for XML parsing. Also, when COBOL is deployed to a runtime container with Web Services capabilities, such as CICS, Business Application services can be perfectly written in COBOL

- Other 3-GL languages

  Depending on the supported languages of the runtime container for the Business Application services components, languages such as PL/I, C++, and C can be considered as well. Technically speaking, even Assembler is still an option and an Assembler program running in CICS and IMS can be Web Service enabled.

- 4-GL languages

  There are many 4-GL tools available from a variety of providers. IBM's premier 4-GL is called *Enterprise Generation Language (EGL)* and can be developed using the *Rational Business Developer* tool. There are few areas with so many heated discussions as the use of a 4-GL. Very simply stated, generally, the big benefit of any 4-GL is productivity gain and the drawback is performance loss. Some programmers will never like using a 4-GL.

Note: Besides the above mentioned mainstream programming models, certain products offer additional specific languages for a specific task. For example, WebSphere Message Broker uses ESQL for modeling mediation and WebSphere Process Server uses BPEL for defining business processes.

A more detailed discussion on this topic will follow in "Language choice" on page 211.

4.2.4 Location

As mentioned in other chapters in this book, a location needs to be chosen for a specific business application service or type of business application service. If assets are being reused, the location of the new service usually will remain the location of the former application component. However, this discussion can get much more complex. First of all, the service interface might not need to reside in the same location as the service component. Refer to *SOA Transition Scenarios for the IBM z/OS Platform*, SG24-7331, for a comprehensive explanation of this topic. Second, one service can consist of multiple components or a service can be a composite service consisting of smaller services residing in different locations. In an enterprise with SOA, Business Application services can reside on various platforms, but an architectural pattern will need to be developed or adopted describing which type of business application service will need to reside where.

A more detailed discussion about this topic is in "Platform and runtime container choice criteria" on page 184.

4.3 Common application patterns

In this section we discuss some of the typical patterns seen for Business Application services on System z. These patterns are important to understand because the characteristics of an existing application influence the approach you can adopt in service enablement. Enabling existing embedded functionality as part of a business application service is an important step in service realization. This section also covers common patterns found when developing new services.

We expect the reader to identify with one or two of these patterns as they match a particular pattern to their environment. These are generally high level patterns. Once patterns have been identified the reader should be able to identify a solution as detailed in the subsequent sections in this chapter.

Note: Where a product such as CICS is used as an example, you could substitute IMS/NATURAL, and so forth.

Common patterns included in this section are:

- 3270 monolithic applications, discussed in "Monolithic applications with 3270 logic" on page 176
- 3270 modular/component applications, discussed in "3270 modular/component applications" on page 177
- Presentation agnostic applications, discussed in "Presentation agnostic business applications" on page 179
- Highly integrated applications, discussed in "Highly integrated applications" on page 183
- "Man in the middle" services (service providers that are also service consumers), discussed in "Man in the middle" on page 183

4.3.1 3270-based business applications

In the following sections we describe 3270 application patterns.

Monolithic applications with 3270 logic

A monolithic application is one that contains many business application components or fragments that, due to the embedded process choreography, is exposed as a complete business process.

Often the nature of these applications was determined by the presentation flow and not based on a common-sense level of granularity for a business service.

Logic within the application is required to ensure the correct screen flow is adhered to. The complexity of marshalling a screen flow within a 3270 application should not be underestimated. This can play a part in understanding the benefits of refactoring over refacing as well as the general ROI expected from an SOA undertaking.

Typically, you should look for these validations and behaviors:

- Has a valid screen flow been compromised?
- Does a particular attribute pass interdependency validation with an attribute from a previous screen?
- Are database updates delayed until screen flow is completed?
- Is data from previous screen flows cached to provide stateful behavior (for example, in IMS SPA, CICS TSQ, and so forth)?

Figure 4-2 shows a typical monolithic 3270 application.

A note on the following diagrams: “P” stands for presentation logic, “B” for business logic, “Dr” for Database read operations and “Du” for Database update operations.

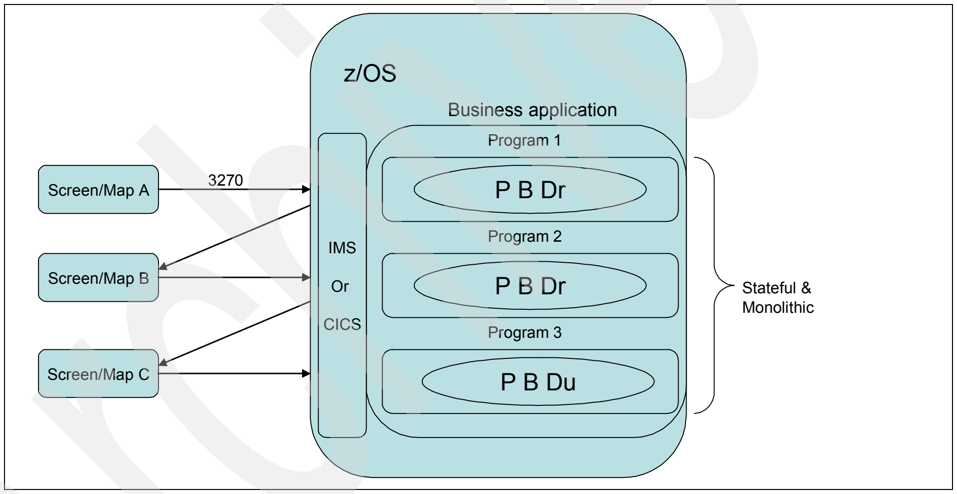


Figure 4-2 A typical monolithic IMS or CICS based 3270 application

3270 modular/component applications

This type of application benefits greatly from a modernization process. The ability to separate components from the application provides reuse and flexibility. The removal of presentation logic also provides a simpler, easier to maintain application. This raises the dilemma of whether to clone the application or to reface it for the remaining 3270 channels. If you suspect you will not be able to address the removal of the 3270 channel aggressively then the reface approach might be best suited. The very nature of providing the service consumer with services in an open standards fashion tends to lend itself towards the aggressive replacement of existing channels.

Factors that might prevent modernization include cost and availability of system and product knowledge.

Figure 4-3 displays a typical modular 3270 application.

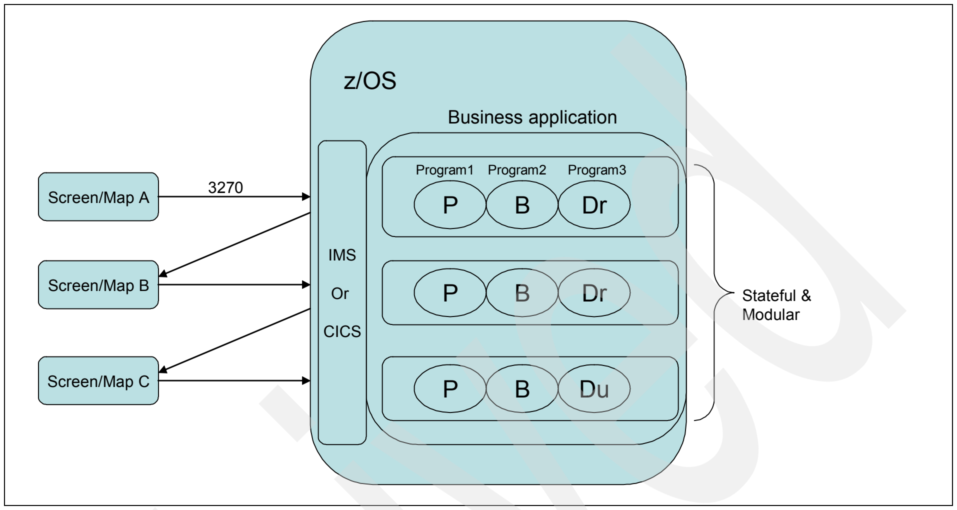


Figure 4-3 Typical modular IMS or CICS based 3270 application

SOA-enabling 3270 applications

Monolithic 3270 applications can hardly be SOA-enabled without any modernization or refactoring. The existing code will need to be modified in most cases in order to reuse parts of the application as services. Modular 3270 applications offer better perspectives for service enablement because these applications have been developed with proper demarcation in mind, and in some cases even multi-channel enablement. SOA enablement activities for 3270 applications can include:

- Separating or eliminating the presentation logic[1].

  In IMS this is rather easy because the IMS business logic and presentation logic are implemented by default in separate modules. Those modules interact with each other through the *Scratch Pad Area (SPA)*. The input and output to the presentation module can easily be redirected to another user interface channel, just skipping the MFS map.

  In CICS the business logic and presentation logic are more tightly integrated.

- Eventually, creating interaction services out of the separated presentation logic. See "Solutions" on page 189 for technologies that can help to do this.

[1] This process is much harder in CICS monolithic applications than it is in IMS.

- Creating services out of the business logic portions of the application. Again, this will be easier and many times quite straightforward for modular 3270 applications. This is the most widely demanded SOA enablement activity.
- Creating data access services out of the data access logic, such as VSAM, DB2, and IMS DB access. Depending on the architecture, a pattern could be to create separate services for data and database access. The benefit of this is clear: being able to reuse the same DB2 query or database update call from different environments and applications. This process can definitely not take place without making modifications to the existing applications because data access is usually embedded in the business logic.
- Moving mediation and workflow logic, if any, from the 3270 application to dedicated ESB and business process services solutions. This is not a very strict requirement for an SOA, but makes sense if such a dedicated ESB and business process services solution is implemented.

4.3.2 Presentation agnostic business applications

In applications that do not contain presentation logic the process of service enablement is simpler than it is for a 3270 type application. This type of application is often referred to as a “COMMAREA only” application in a CICS scenario. In an IMS environment it does not really matter that much whether an application has a 3270 component or not. The request can be sent to the business logic program and the request comes back directly from the business logic program, even if an MFS map exists. The MFS is only used if an end user uses a 3270 terminal to drive the MFS map.

Nowadays, most z/OS applications without 3270 presentation logic consist of reusable business logic modules that are being used by other z/OS applications or driven by external user interfaces, usually a Web interface. However, there also are monolithic applications without 3270 presentation logic.

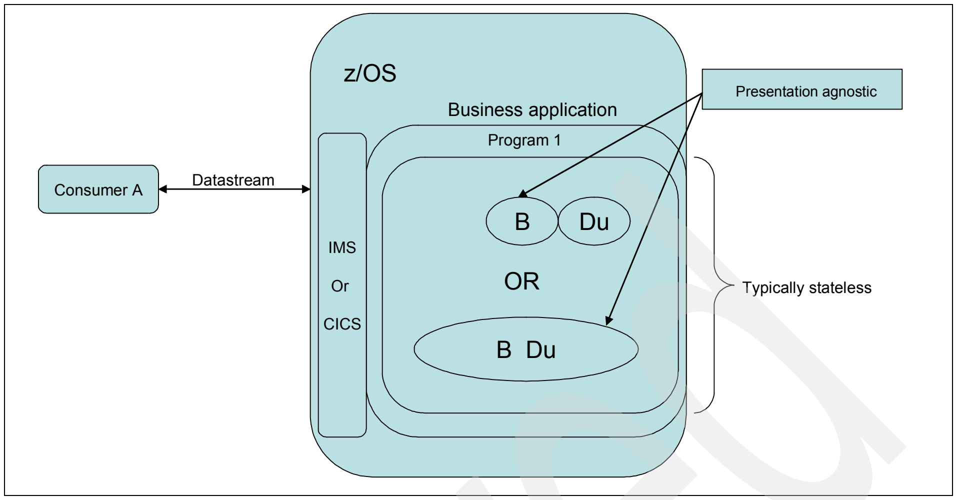


Figure 4-4   3270 application without built-in 3270 presentation logic

SOA enabling applications without presentation logic

SOA activities for these types of applications can include:

- Creating services out of the business logic portions of the application. This is the most widely demanded SOA enablement activity.
- Creating data access services out of the data access logic, such as VSAM, DB2, and IMS DB access. Depending on the architecture, a pattern could be to create separate services for data and database access. The benefit of this is clear: being able to reuse the same DB2 query or database update call from different environments and applications. This process definitely cannot take place without making modifications to the existing applications because data access is usually embedded in the business logic.
- Moving mediation and workflow logic, if any, from the application to dedicated ESB and business process services solutions. This is not a very strict requirement for an SOA, but makes sense if such a dedicated ESB and business process services solution is implemented.

Even with the complexity of presentation logic removed you might still be presented with a certain amount of necessary component refactoring depending upon the nature of the current application. In some cases an application might still be structured based on some previous screen flow. Conversely, you might find the application to be a monolithic application that contains all business logic for a complete screen flow. In this case you might be able to service enable it relatively easily. The service enablement of this monolithic application without component refactoring does, however, influence the granularity level of the service. This tendency towards a *courser grained* service is acceptable on the understanding that a certain amount of process choreography will always be embedded within the application service and not abstracted into something like BPEL with a business process service. While there is nothing inherently wrong with course grained services, a process should be undertaken to identify all functionality within the components of a service and a decision made as to whether this tightly coupled process choreography will affect the level of flexibility on offer to a business. After all, one of the foundation stones of an SOA is the ability to choreograph and compose Business Application services into complete business processes.

To demonstrate this we consider the example of a banking application. This application will retrieve the available balance of all of a customer's accounts. Let us assume the existing application contains logic to retrieve balances but also includes a process to charge the client for this service. The first consumer of this service might actually find this tightly coupled choreography of benefit. However, a subsequent consumer might not require the charge functionality. When multiple apparently different business activities or functions are packaged in one service it is just a matter of time for a consumer to come along that only needs part of the functionality in the service.

This example is meant to illustrate that when attempting service enablement of existing legacy applications, there are many factors that influence granularity levels of the service. This should be carefully considered when deciding on your architectural approach to service enablement.

Figure 4-5 on page 182 shows that the business logic portion of an existing application can contain a number of functions that could all be potential atomic services. In the analysis stage of an SOA project, you must decide which of those portions actually need to become reusable services. Note that it is even possible that one program results in multiple services. Tooling exists to help in this "mining" process.

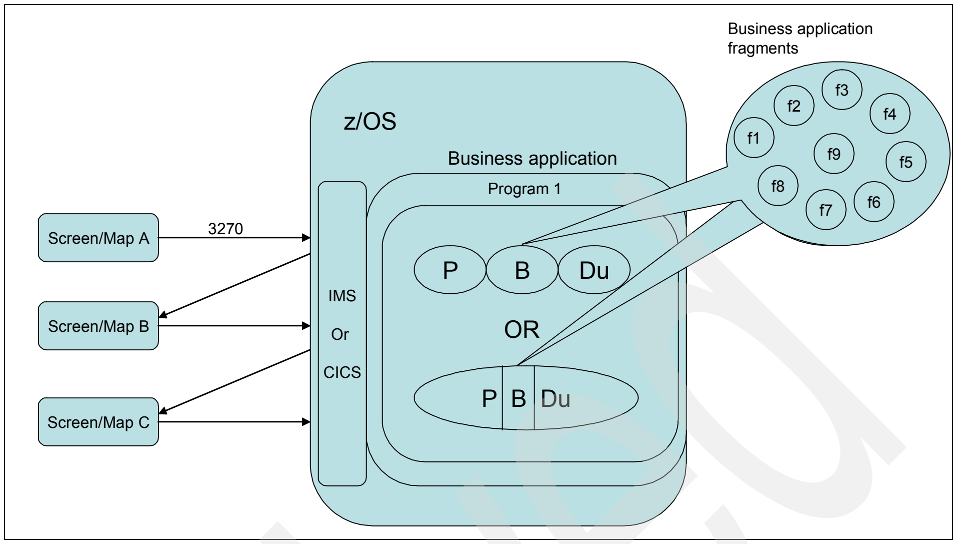


Figure 4-5 Refactoring of existing code fragments: Mining process

Figure 4-6 shows the application logic, but split up into desired services. The diagram also shows business processes driving the individual Business Application services.

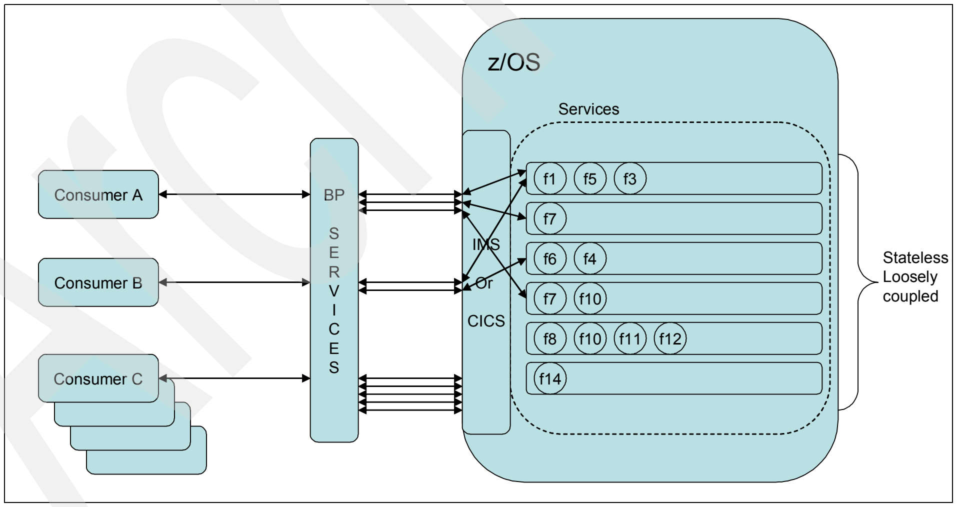


Figure 4-6 Refactoring existing code fragments: Deployment process

4.3.3 Highly integrated applications

The degree to which existing applications or services within a business process require integration to other applications will influence the architectural decisions and the resulting landscape of the solution. Tight integration to existing applications, such as a CRM application based on IMS or CICS, might require integration at a module or class level as opposed to a more course grained integration at, for example, the service level (assuming your application has been service enabled). Should you require this level of integration, you should co-locate the service within the same container (IMS, CICS, and so on). Should you not require this level of integration you might be able to provide integration using SOAP, which offers more flexibility of location.

4.3.4 Man in the middle

A common scenario within an SOA implementation is one where the service provider is also a service consumer. In other words, the service that gets called by a service consumer can also make service calls itself to another service provider. This process of “chaining” services is not uncommon in an SOA and might go well beyond just two stops.

It is important to understand how often an IMS- or CICS-based application previously and now requires the ability to either synchronously or asynchronously invoke a service hosted elsewhere. Both IMS and CICS applications provide the capabilities to offer this service “call out” feature. The suitability of such environments to play the role of client or consumer needs careful consideration. A JEE/WebSphere Application Server model is possibly better suited to this situation because of the ability to dynamically expand and contract a thread pool and the resources required by each thread. To a certain extent CICS offers this feature in its ability to sustain multiple TCBs within a region. The IMS model is somewhat different, with a static definition of threads/Message Processing Regions (MPRs). Be very careful with directly invoking an asynchronous service from a CICS or IMS program because the consumer program goes into wait status for some time.

A common error here is basing the architectural decision on average response times or an SLA agreement between consumer and provider. The reality is that a worst case scenario should be considered when modelling the number of CICS TCBs or IMS dependant regions required to maintain a specific Quality of Service.

Consider this practical example:

A service with an average response time of 1 second to a synchronous request might only require 10 IMS dependant regions to prevent queue build up and a

QoS reduction. However, if the service provider suffers failure and the IMS-based service requester is prepared to wait up to 5 seconds for a response before a time out is signalled, the system impact should be understood. Possible QoS failures should be considered as part of the architectural decisions.

4.4 Platform and runtime container choice criteria

As the section name suggests, there are two decisions to be made:

- Which container should be used to deploy the Business Application services?
- Which platform should the container run on?

The preferred programming model should also be a consideration here.

To be able to make these decisions, we have to know what kind of Business Application services we are dealing with and what the expected QoS is. The following sections address these topics.

4.4.1 Business Application service types and characteristics

Very often the characteristics of a service or its non-functional requirements play a role in determining where they can be deployed. Here we discuss some of the requirements and characteristics typically found in Business Application services:

- High volatility
- High data activity
- Resource intensive
- High reusability
- Security
- High throughput

High volatility

Services might change very often and there might be a requirement to offer multiple versions of a service to different consumers at the same time. Here an ESB should be used in combination with a service registry to provide abstraction and mediation to support multiple consumers of different versions of a service. This information is vital in impact analysis and change management if you envision a highly volatile service.

High data activity

Some services require access to large volumes of data or are typically data bound with high data access levels. This can be inquiry or update. If this is existing data it could be in the form of an IMS DL1 database, VSAM or QSAM files, or a DB2 database. Proximity to this data might influence your architectural decisions. There might be value in considering a DB2 stored procedure where proximity of business logic and data is crucial. A DB2 stored procedure can be service enabled by itself.

Resource intensive

Certain services might be resource intensive while others are not. It is good to understand how resource intensive each service is and what type of resources are heavily consumed. Is it CPU, memory, or I/O? In some cases, the resource intensity might influence the choice of the programming model, the runtime container and ultimately the hardware platform.

For example, a service becomes CPU intensive if many calculations have to be performed or there is much XML parsing going on. Also, encryption can be very CPU intensive if not offloaded to the hardware. CPU intensive services should run on hardware with fast CPUs.

A service becomes memory intensive if there is a lot of data processing and caching going on. The longer it takes to run a transaction or request, the more likely it is that memory allocation increases. Note that certain runtime containers and programming models do an excellent job of memory management and others not. Memory intensive services should be deployed to a runtime container with excellent memory management capabilities.

I/O intensity depends on read and write operations on data stored on disks. Obviously, an I/O intensive service should be located close to the data and in a runtime container with excellent database attachment facilities or connectors.

High reusability

While the purpose of service provisioning is to maximize reuse of a service in as many processes as possible, you will find some services are likely to be used more often than others. In many environments this is often a customer inquiry type service that has very high reuse. The amount of reuse and the location of peer services that might form a complete business process can influence your architecture.

Security

Common security requirements with respect to Business Application services include, but are not limited to:

- Authorization to use a Business Application service based on user ID or security context of incoming request
- Access control from within the Business Application service to other resources, for example, a file, a data set, or a database, based on user ID or security context
- Security administration functions, such as querying whether a certain user is allowed to perform a certain database update

High throughput

You might find certain services have high throughput requirements. This is often manifested by the nature of the process. The QoS of the runtime container and platform need to support the throughput anticipated.

4.4.2 Business Application services QoS

In this section we consider QoS issues as they relate to Business Application services.

Availability, recoverability, and reliability

Core business processing is now required to be available 24x7. Business Application services contain core business logic and must have a relatively high availability. Mission-critical Business Application services must have a failover and disaster recovery implementation. In fact, this means that the runtime container must have at least one active duplicate instance running somewhere else and that workload can be directly routed to the other instance (or instances) in case of outages or congestion. Hot standby instances of the runtime container might be necessary too, depending on the requirements. It is a big advantage if the runtime container can be clustered into a single image with multiple physical nodes. This can be done with all IBM transaction managers on z/OS, including CICS, IMS, and WebSphere Application Server. Parallel Sysplex can be used for real-time data sharing, so that all physical nodes work with the same data at any point in time. *Dynamic Virtual IP Addressing (DVIPA)* can be used to access different nodes from outside based on one virtual IP address. This way the clients and consumers of services do not need to worry about which physical node to access.

Accounting

In a charge-back model the calculated costs of a business process are charged against a particular business unit or Line Of Business (LOB). Traditionally, a business application would be defined by the transaction code it executed under within the specific TP Monitor (CICS or IMS). An SOA model alters this. The functionality that is associated with a particular transaction is now a service or collection of services. Services are designed to be applicable to a multitude of business processes and this is where charge back ambiguity arises. This requires specific data to be logged to aide charge back. This could be a client IP address, a user ID or a unique application identifier. The System z platform provides many mechanisms to log data that might be used in a charge-back model. Both the *System Management Facility (SMF)* and the logstreams facility offer a very high speed and fault tolerant solution to this problem. Tools are available that use the SMF records as a source for reporting.

Transactionality

Especially with Business Application services, transactionality requirements might come into play. The requirement for *2-Phase Commit* arises when multiple data updates are being performed that are dependent on each other. In many businesses, just one failure can have disastrous consequences. Maybe you have run into one of those situations where you went to an ATM to make a withdrawal and the machine failed in the last seconds to give you the money, but later on you noticed on your bank statement that the money has been actually taken out of your bank account? The cause of this specific problem is that the update on your bank account was initiated before the successful end of the transaction, which would have been handing over the money.

In an SOA, we might see requirements for situations in which multiple services belong to one transaction, either being executed in parallel or sequentially. Those multiple services might all perform data updates. It is also possible to have a single service with multiple data updates.

A note on loosely coupled services and transactionality: In an SOA everybody should strive to create atomic, loosely coupled services with a realistic granularity. The goal is to reach maximum reusability, but not at any cost. In some cases you might see clusters of services that must be executed as one transaction or unit of work.

If these transactionality requirements exist, the behavior of the runtime container becomes very important. There are several ways to deal with 2-Phase Commit transactions:

- Let the runtime container deal with the commits or rollbacks depending on the situation. The runtime container can use O/S functions to do this.

In the traditional environments on z/OS, the assumption is that the transaction monitor carries the responsibility for reliability, not the application. An application developer needs to think carefully about where the transaction begins and ends, but the transaction monitor is in control of the execution and takes the decisions on commit and rollback. On z/OS, in case of a rollback, the transaction monitor backs out updates up to the last sync point using logs.

- Model and develop compensation transactions that get executed in case of failures.

  In many newer runtime containers, the vision of a transaction monitor implicitly managing the 2-Phase Commit is different. The developer must explicitly configure what needs to be done in terms of a 2-Phase Commit, if supported at all.

The System z platform is unparalleled in its ability to provide atomic transactionality both locally and across distributed platforms. The key to this is the ability for a 2-Phase Commit (2PC) to be coordinated with TP monitors (TPM), Database Management systems (DBMS) and messaging middleware. Because SOA introduces the ability to coordinate services on disparate platforms into complete business processes, the need to coordinated an “all or nothing” approach is vital. This is termed an “atomic transaction.” This global transactionality across distributed platforms is commonly implemented by applying the XA Transactionality standard. This is a standard proposed by the X/Open standards group. For more information see *Distributed Transaction Processing (X/Open Cae Specification S),* published by The Open Group, ISBN 1-87263-024-3. The z/OS system provides this functionality in the form of the Resource and Recovery System (RRS).

As far as atomic transactionality and services are concerned, the OASIS standards consortium has proposed the WS-AtomicTransaction. This specifies a model for distributed Two-Phase Commit using SOAP-based Web Services. WS-BusinessActivity defines an asynchronous model for compensating failed processes using undo actions to reverse the effects of individual steps of the process.

Further information regarding WS-AtomicTransaction and WS-BusinessActivity can be found at the following locations:

- WS-AtomicTransaction specification
  `http://www.ibm.com/developerworks/library/ws-atomtran/`
- WS-BusinessActivity specification
  `http://www.ibm.com/developerworks/webservices/library/ws-busact/`
- Transactions in the world of Web Services, part 1 and part 2
  `http://www.ibm.com/developerworks/webservices/library/ws-wstx1/`
  `http://www.ibm.com/developerworks/webservices/library/ws-wstx2/`

4.5 Solutions

In the following sections we take an in-depth look at the approaches to creating Business Application services from both new and existing business components. We then discuss the services and tooling available to assist in your chosen approach. The solutions discussed here do not constitute a comprehensive list; they were included because we believe them to be best of breed to address the common patterns identified previously.

4.5.1 Service-oriented modeling and architecture (SOMA)

Regardless of whether you are refactoring or replacing a business application, the adoption of a solution development methodology is highly recommended. *service-oriented modeling and architecture (SOMA)* is one such methodology that can assist you in identifying, designing, and developing SOA application components.

The SOMA process consists of three general steps: identification, specification, and realization of services, components, and flows (typically, choreography of services). This is illustrated in Figure 4-7.

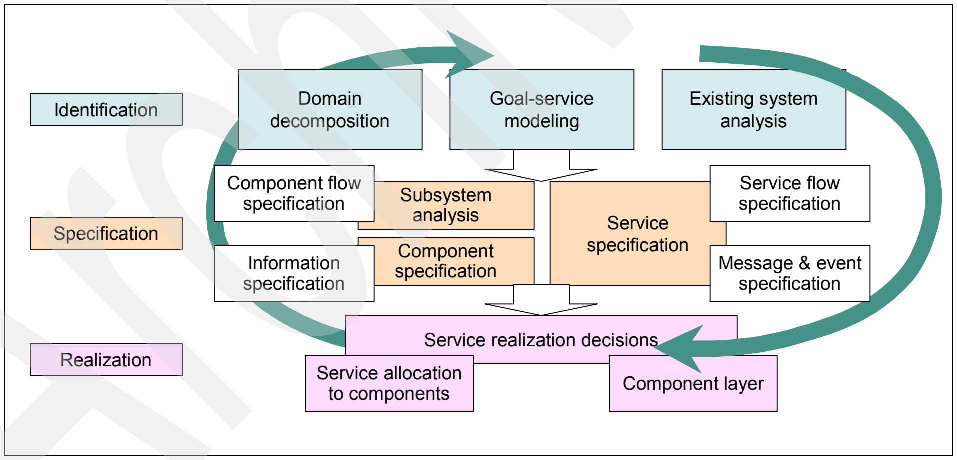


Figure 4-7 The SOMA process flow

Service identification

The goal here is to reveal the existing business processes broken down into functional components. The completeness of service identification relies on a top-down, bottom-up, or middle-out analysis process being undertaken. The top-down approach is used during domain decomposition, whereas the bottom-up approach is used during existing asset analysis. The middle-out approach is referred to as goal to service modeling and provides the final sanity check. Applying this final process can ensure business goals are being met and that reuse and flexibility are truly achieved by the services identified.

Domain decomposition is a process of extracting functional areas and processes from the business domain model. Producing high level use cases can assist in service identification. This process assumes that a level of as-is documentation or understanding of the business domain exists. Should this not be the case, as-is analysis might be required to understand existing business processes.

Existing asset analysis is a bottom-up approach whereby business functions are extracted from existing logic, screen maps, and documentation to form components. These components might in turn be repackaged as a business application service.

Goal-service modeling is a middle-out approach used to ensure that any services identified align with the business goals and provide the correct granularity level to ensures business flexibility through choreography.

Service specification and allocation

This topic is not covered in this chapter because neither of these steps in the SOMA process are specific to System z. Refer to the following Redbooks publication: *SOA Foundation Service Creation Scenario*, SG24-7240.

Service realization

The service realization process relates to the actual implementation of a service. Within the context of the System z environment, the realization process must consider the following alternatives:

- Reface
  Wrap existing business logic that will continue to contain presentation logic.
- Rejuvenate
  Compose a service from refactored business application components and expose the service via an open standard such as SOAP.
- Replace/Build
  Compose a service from components that are developed from new business requirements or requirements that have been derived from a bottom-up approach from an existing application.

4.5.2 Reface, rejuvenate, or replace?

Many factors will influence this choice. Among them are:

- Budget
  What funding is available? The cost for modernizing an application might vary greatly depending on the SOA approach and the technology deployed.
- Appetite
  What is the attitude towards change within the organization. Does the organization believe SOA has a good ROI?
- Alignment
  Is the SOA process being driven symmetrically by both business and IT?
- Skills
  What skills are available with respect to both the existing technology (such as COBOL, CICS, and IMS) and the new technology to be used (such as Web services, ESBs, and Web technologies)?
- Priorities
  Would competing business initiatives have higher priorities than SOA application enablement (rejuvenation or replacement)?
- Scope
  As-is or additional functionality: Is the SOA enablement of an existing component satisfactory for business or is additional functionality required?

Now that we have discussed the key architectural decisions as well as some common patterns found within existing applications on System z, you should have a better understanding of SOA enablement of business applications. This should assist in find the approach that best suits your situation.

4.5.3 Solutions for refacing applications on System z

Refacing of an application refers to the process of applying some form of transformation via middleware to provide a new user interface to an existing application. This transformation might, for example, transform a 3270 datastream into HTML or XML. In this scenario the consumer presents a 3270 datastream to an IMS or CICS application and in turn a 3270 datastream is returned. This 3270 charade allows the refacing of an application, but the underlying application remains unchanged. This is beneficial if both a true 3270 channel as well as the refaced channel must be maintained in parallel. Thanks to refacing middleware, a single application can service two or more delivery channels without modification. Middleware such as WebSphere Application Server can then in turn expose this application as a Web service.

At first glance this might seem like a satisfactory solution. However, the fact that the existing application remains unchanged does not bring us much closer to achieving the SOA benefits. The limitations of this approach are:

- No service scoping has been done, so the scope of the component is the same as what it was before.
- Reusability will not be optimal, because again, the scope of the component that has been rejuvenated is so specific that reuse in other applications, business processes, and business units is not very likely.

Providing very course grained services reduces the level of flexibility and therefore might reduce the likelihood a service can be reused.

One of the main benefits of SOA is the ability to expose services whose level of granularity has been carefully considered. This in turns allows services to be choreographed into business processes. This ability is fundamental to SOA. The exposing of functionality via a Web Services interface might in no way contribute to the establishment of a service-oriented architecture. However, if the granularity level of the resulting service provides the necessary flexibility and reuse, the refacing of an application might be an appealing choice.

We look at four different solutions to provide application refacing. These are:

- Rational Host Access Transformation services (HATS)
- CICS Service Flow Feature
- IMS MFS Web enablement
- IMS MFS Web services

Rational Host Access Transformation Services (HATS)

Rational Host Access Transformation Services (HATS) provides both development and runtime components to assist in the transformation of 3270 type applications to have a variety of interfaces. The HATS development tooling is provided as an Eclipse plug-in to Rational Application Developer (RAD). RAD allows you to reface the application as a Web interface or as a Web Services interface. In addition, the ability to build portlets to be deployed on WebSphere Portal was recently announced.

Importing BMS or MFS maps is the initial starting point in application refacing within RAD. From these 3270 maps you are able to select input and output fields as well as the mapping of any user interactions.

The process is shown in Figure 4-8 on page 193.

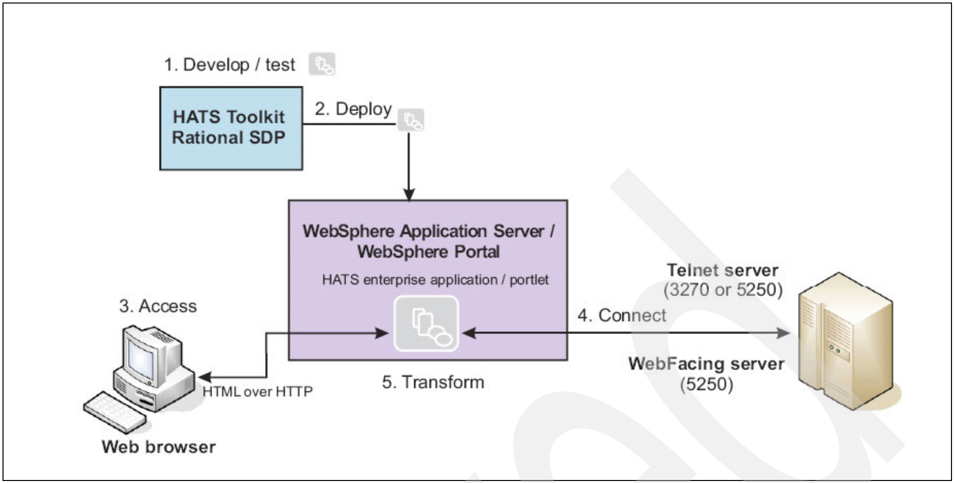
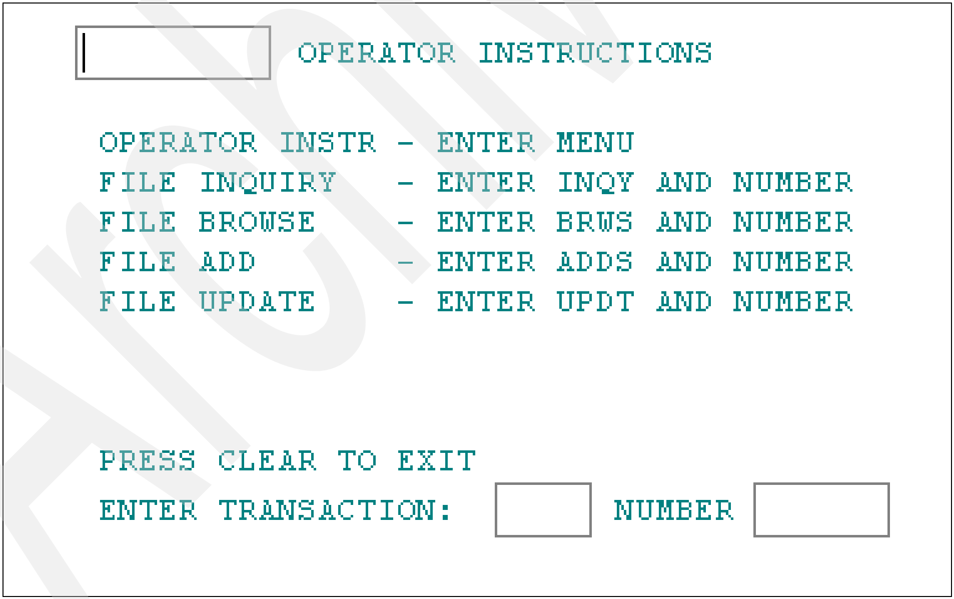

1. Develop / test
HATS Toolkit Rational SDP
2. Deploy
WebSphere Application Server / WebSphere Portal
HATS enterprise application / portlet
3. Access
HTML over HTTP
Web browser
5. Transform
4. Connect
Telnet server (3270 or 5250)
WebFacing server (5250)

Figure 4-8 Refacing with HATS

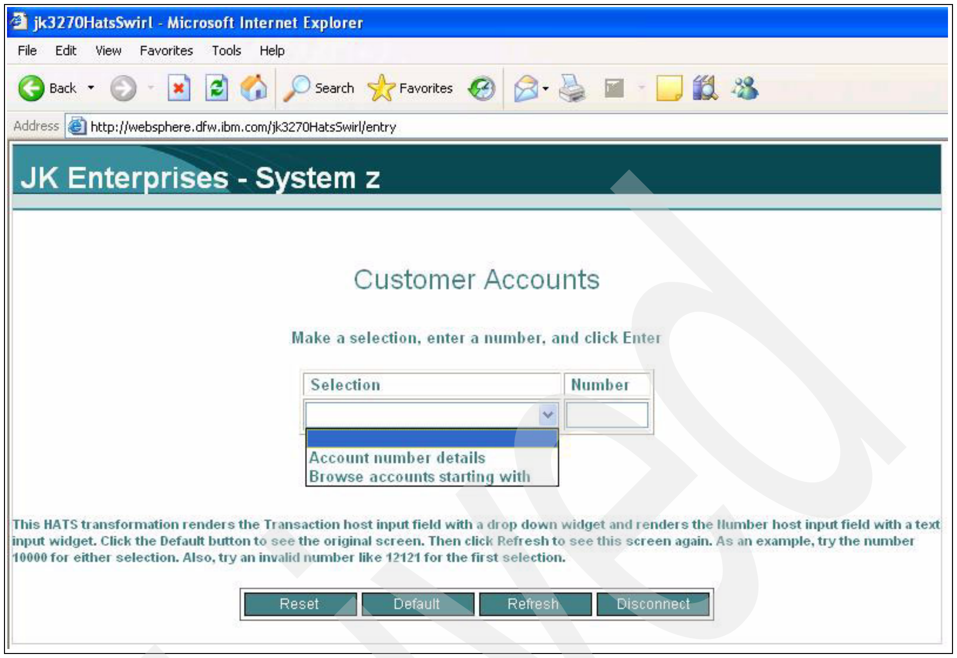
The next two figures provide an example of how an existing 3270 application is refaced to have a Web (HTML) interface.



```
              OPERATOR INSTRUCTIONS

  OPERATOR INSTR - ENTER MENU
  FILE INQUIRY   - ENTER INQY AND NUMBER
  FILE BROWSE    - ENTER BRWS AND NUMBER
  FILE ADD       - ENTER ADDS AND NUMBER
  FILE UPDATE    - ENTER UPDT AND NUMBER



  PRESS CLEAR TO EXIT
  ENTER TRANSACTION:        NUMBER
```

Figure 4-9 A typical 3270/character-based application

In the example in Figure 4-10 on page 194 the options available via the INQY and BRWS transactions are enabled via a drop-down list.


jk3270HatsSwirl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://websphere.dfw.ibm.com/jk3270HatsSwirl/entry

JK Enterprises - System z

Customer Accounts

Make a selection, enter a number, and click Enter

Selection	Number
Account number details Browse accounts starting with	

This HATS transformation renders the Transaction host input field with a drop down widget and renders the Number host input field with a text input widget. Click the Default button to see the original screen. Then click Refresh to see this screen again. As an example, try the number 10000 for either selection. Also, try an invalid number like 12121 for the first selection.

Figure 4-10 A new HTML based interface

A next step that can be taken with HATS is to create a Web service. You can then create an architecture as shown in Figure 4-11. The “S” represents a Web Services interface.

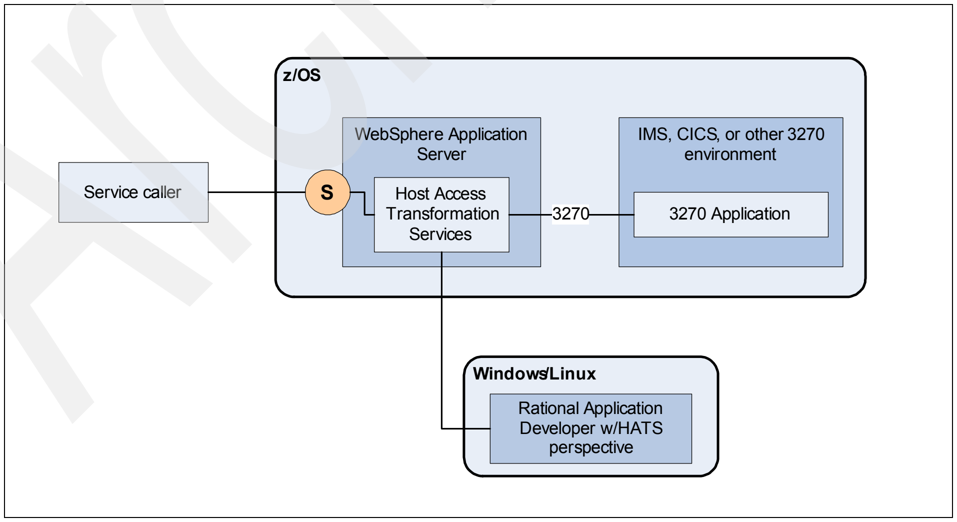

z/OS
WebSphere Application Server
IMS, CICS, or other 3270 environment
Service caller
S
Host Access Transformation Services
3270
3270 Application
Windows/Linux
Rational Application Developer w/HATS perspective

Figure 4-11 HATS with a Web Services client

HATS offers much more functionality than just converting legacy data streams to a Web interface, but in this section our aim is only to mention the “refacing” functionality.

For further information refer to the Rational Host Access Transformation Web site at:

```
http://www.ibm.com/developerworks/rational/products/rhats/
```

CICS Service Flow Feature

CICS Service Flow Feature provides the capability to aggregate existing CICS programs into composite business services, which can in turn form part of a business process. The ability to aggregate several CICS-based screen flows into one complete business application service is useful in providing course grained services. CICS SFF fits in both a refacing and rejuvenating approach. CICS SFF allows you to eliminate 3270 screens, but also creates an opportunity to better componentize your application, as well as creating the proper service granularity. However, you have to keep in mind that aggregation of CICS programs into a service works well, but decomposition of programs into smaller services cannot be done without a proper refactoring approach and tools.

The diagrams in Figure 4-12 and Figure 4-13 on page 196 illustrate the difference between a CICS application before and after using CICS SFF.

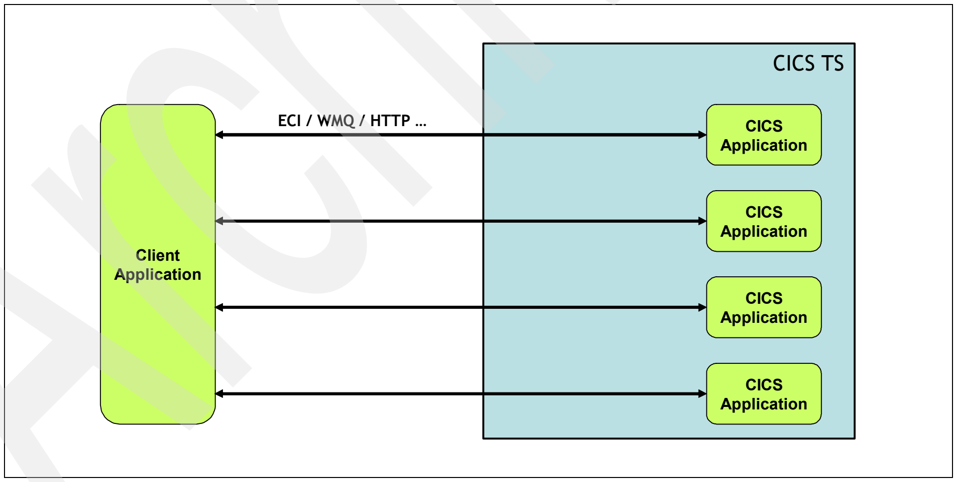


Figure 4-12 Traditional external access to CICS

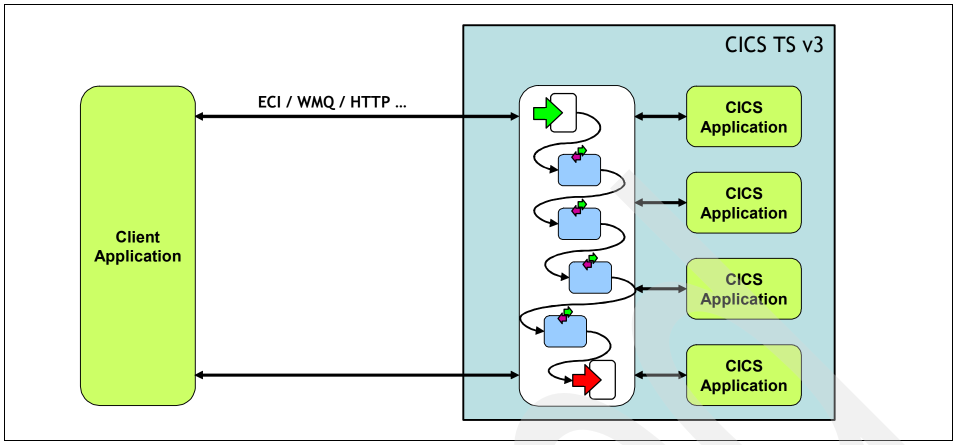


Figure 4-13 A composite CICS Web service after using CICS SFF

The aggregation of these programs into a usable service is necessary because it is very unlikely that the functions contained within a single program would provide a complete business component. Even if the functions within a program were stateless, they are probably too fine grained to provide a complete service. Therefore the need for aggregation. You might find, however, that once aggregation is complete the converse applies. This is due to the fact that the existing programs are likely to contain some embedded business process choreography resulting in a course grained service. A service can become so course grained that its flexibility and therefore the likelihood of its reuse is reduced or eliminated.

In cases where the luxury of functional decomposition and rejuvenation is not available, the CICS Service Flow Feature is an extremely useful tool in orchestrating functions across multiple CICS applications into usable business application components. An added benefit is that the Quality of Service delivered by CICS and the System z platform is in no way compromised.

The logical structure of a flow designed with CICS SFF is shown in Figure 4-14 on page 197.

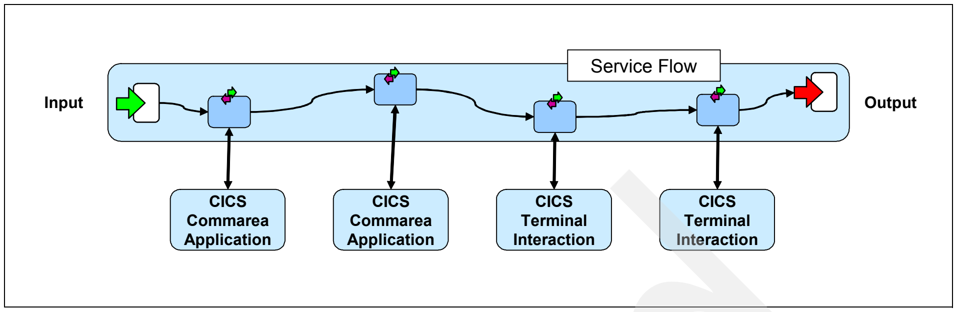

Service Flow
Input
Output
CICS Commarea Application
CICS Commarea Application
CICS Terminal Interaction
CICS Terminal Interaction

Figure 4-14 A flow in CICS SFF

CICS Service Flow Feature (SFF) consists of two components:

- The *Service Flow Runtime*, which is a no-charge separately orderable feature of CICS TS 3.1 and later.
- The *Service Flow Modeler (SFM)*, which is included in WebSphere Developer for System z Version 7 and Rational Developer for System z (RDz).

The overall deployment model of CICS SFF applications is shown in Figure 4-15.

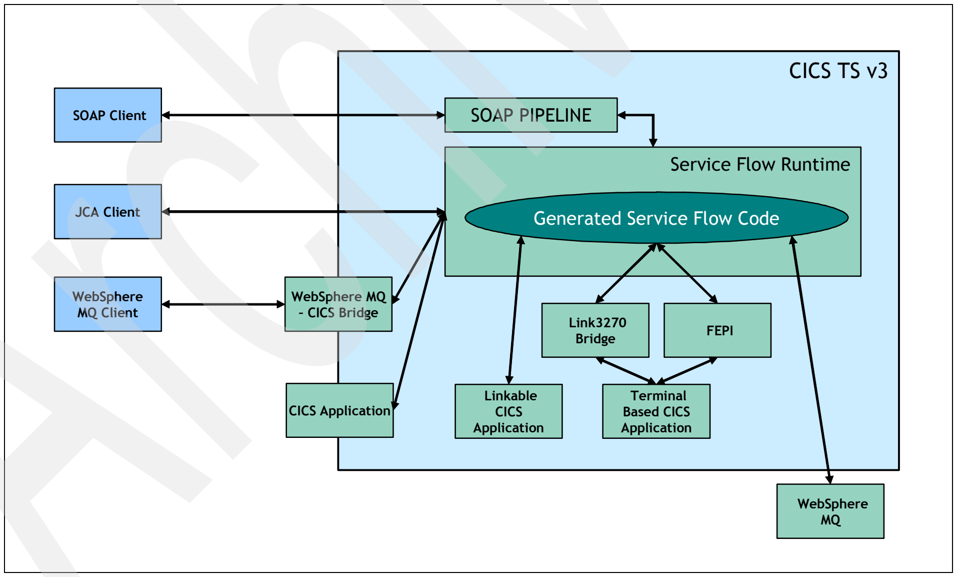


Figure 4-15 Overall deployment model of CICS SFF

The overall service flow and any relevant decision point must be described within the Service Flow Modeler. SFM allows a developer to define flows, nodes, and message mappings graphically. An example of a flow is displayed in Figure 4-16.

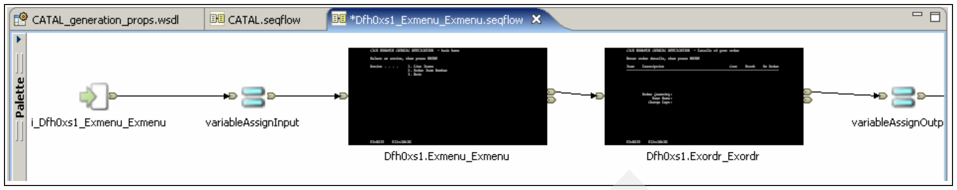


Figure 4-16 CICS Service Flow Modeler in RDz

The SFF allows both business logic and 3270-based programs to participate within a service flow.

IMS MFS Web enablement

This feature allows an existing MFS-based IMS application to be refaced with an HTML-based presentation layer.

The development time tooling comes in the form of the stand-alone MFS Web enablement utility. This utility allows for the importing of MFS source, which in turn produces Java classes and XML Metadata Interchange (XMI) files. The resulting packaged WAR should then be deployed to a WebSphere Application Server.

At runtime the WebSphere Application Server application using the MFSn instance servlet, MFS servlet, and MFS adapter will allow for the conversion of a HTML Web page to a 3270 bytestream understood by the IMS MFS-based application. In the response the bytestream is then used in the generation of a dynamic Web page. This is illustrated in Figure 4-17 on page 199.

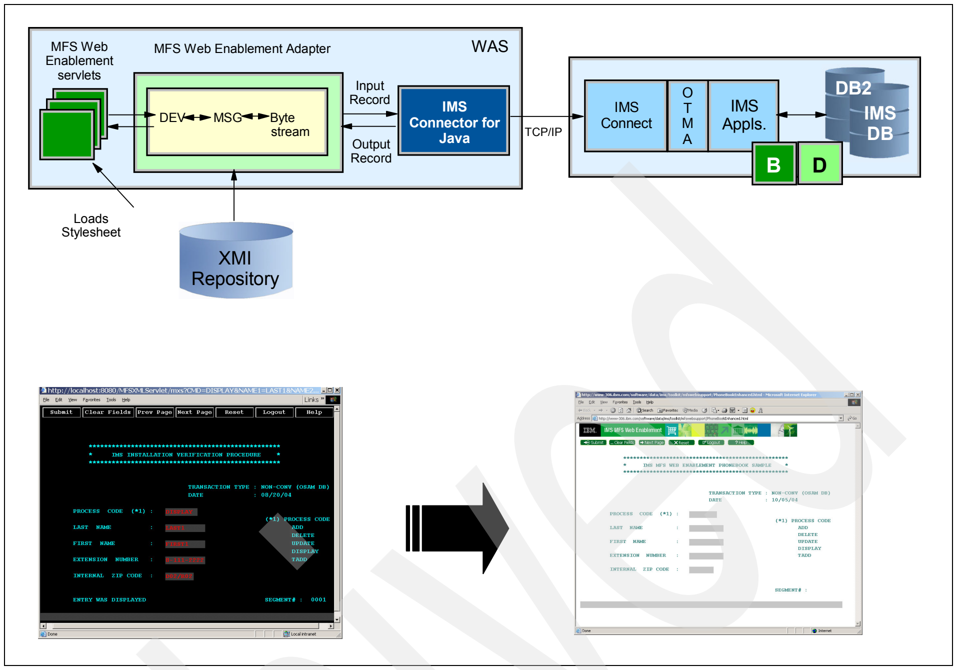


Figure 4-17 Application refacing using IMS MFS Web enablement

By using IMS MFS Web enablement we are able to easily reface the existing application with a Web “front-end” without changing the application. This certainly improves the look and feel of the application. What it does not do, however, is provide any form of reusable component. Although this process might have some appeal, it should be noted that in terms of SOA enablement, it contributes very little.

IMS MFS Web services

This feature allows an existing MFS-based IMS application to be exposed as a SOAP-based service.

The development time process starts with the importing of the MFS source maps. If the MFS source is not available, it can be recreated using the IMS MFS Reversal Utility. For more information, see *IMS Message Format Services Reversal Utilities for z/OS User's Guide*, SC27-0823. Once the MFS has been imported into RDz you are able to generate the WSDLs that describe this service. As with the MFS Web enablement process, an XML metadata interchange (XMI) file is generated. This metadata will be used at run time to perform the transformation between XML and bytestream as expected by the IMS application.

Figure 4-18 shows both IMS MFS Web enablement and IMS MFS Web services.

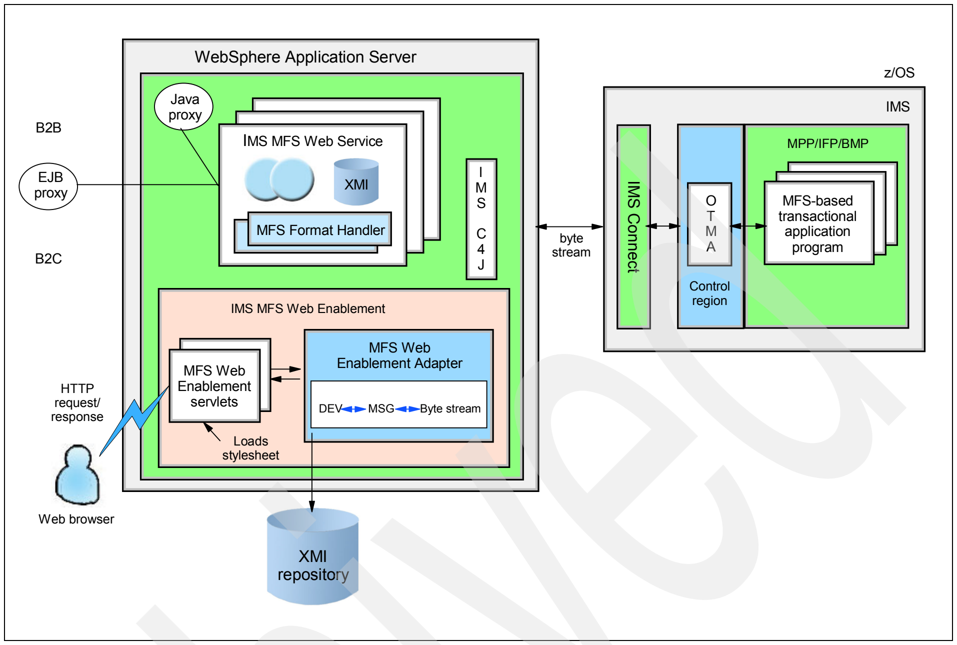


Figure 4-18 SOAP enablement of an existing IMS MFS based application

Other solutions for refacing

We discussed four possible solutions for refacing a traditional application on z/OS. There are more solutions for refacing, from a number of companies other than IBM. Some organizations have built their own frameworks to perform refacing. The key objective of all of those solutions is to be able to work with a native datastream, such as 3270, in a J2EE application server environment. Once that is accomplished, anything can be done in the J2EE server based on that datastream, creating a Web service interface, a portlet in WebSphere Portal, a human task as part of a WPS business process, and so on.

4.5.4 Solutions for rejuvenating applications on System z

In many cases rejuvenation is the way to go if you wish to get to an SOA. We now discuss the challenges, the solutions, and the tools pertinent to the rejuvenation approach.

The challenge of rejuvenation

In this section we discuss legacy rejuvenation with respect to Business Application service enablement. Often referred to as legacy modernization, there are many misconceptions and approaches to this wide-ranging topic. Let us first discuss what we understand by the term legacy rejuvenation.

Legacy rejuvenation within the SOA context relates to the process of service enablement of an existing business application. By service enablement we are *not* referring to SOAP enablement. SOAP might be the mechanism by which we choose to expose a service, and how the resulting service is exposed to potential consumers, but here we are referring to the process of taking an existing application, with its many embedded business processes and functions, and refactoring these individual components into a Business Application service. This refactoring process must produce services with the correct level of granularity in order to provide a high level of flexibility and reuse. This process of mining business functions from existing applications and their reassembly into a service is the fundamental building block of legacy rejuvenation.

In simple cases you might find the existing application contains exactly the correct functions to provide a business service. Conversely, you might find the application is too course grained and if it were to be service enabled its likelihood of attracting many consumers is low.

SOA literature tends to warn of the pitfalls of services being too fine grained, often referring to them as “chatty” services. However, when dealing with legacy business applications on z/OS, the application itself is often far courser grained by nature than one that might have been implemented using an object-oriented language. Because legacy applications are usually monolithic and based on some screen flow, their granularity tends to differ. This is often to such an extent that your existing application might be a service candidate as it is. It is still, however, extremely important to undertake a rejuvenate by refactoring process. This is necessary to understand what business functionality exists currently within the application and to model this to service requirements. Another by-product of the refactoring process is understanding what hard-wired process choreography is coded within an application. Once this has been identified a decision can be made on how this choreography will be implemented in the SOA landscape.

Given the need to refactor the application, we need to adopt a method for understanding and documenting the functions that have been mined from the application. To demonstrate how such a process might look let us take a very simple method. Once we have identified the application we print the program listing and spread the pages in the listing out on the floor. We then identify what we believe to be business functions/fragments and cut them out from the relevant pages. We can then file these functions once we have provided some form of name and function description to each one; an index might also be useful. Clearly this is a silly example, but it serves to show the concept that should be employed. RTW allows for this exact functionality, but with the capabilities of organizing, sharing, collating, and so forth, of the refactored logic.

The business functions identified can now be used to form Business Application services. The removal of presentation logic results in a simpler application and offers major benefits over the application refacing approach, in particular that when refacing is chosen over rejuvenation the 3270 process model is further entrenched. In the rejuvenation approach applications no longer contain complex logic to marshal screen flow or delay interdependency validation until screen flow is completed. This new application is therefore presentationless and stateless. We now have a simpler situation whereby all data is presented to the service via a request and (typically) all data is returned via the response. Although the application is now simpler we believe that a 3-tier approach is still necessary. It might seem contradictory that we propose a 3-tier model for what is a presentationless application, but even though the application is now presentation agnostic it is not TP monitor agnostic.

We believe the best approach is to place the TP-specific calls in what we traditionally term the "presentation tier." This is typically a GU/GN call to the IOPCB in IMS and a RECEIVE/SEND in CICS. This enables business functionality to be modularized into a separate business tier. The flexibility of a multi-tiered model can be important if a multi-channeled application is required. It also allows a certain amount of process choreography to be implemented within the z/OS platform. We are not suggesting a BPM functional replacement, but rather the flexibility to provide services that are less chatty than would be ideal should performance be an issue.

The business application layer is obviously now composed of refactored functionality. This business application tier is typically far simpler than the application it might have been part of. All screen marshalling and state persistence is now removed. The third tier is database access, but because this does not influence the SOA enablement to any large degree it is beyond the scope of this publication.

The scenario we now have is a business application tier that is modular in nature and allows multiple channel access. The presentation tier shields the business logic from any TP Monitor specific syntax and being modular allows for its inclusion in business process choreography. We believe this to be an ideal SOA enablement legacy rejuvenation strategy.

Legacy rejuvenation to achieve SOA enablement will incur a certain amount of overhead. This cost is mitigated by the use of tooling (discussed in the next section) and further offset by the fact that during refactoring the business logic fragments remain intact and therefore do not require as full a testing cycle as would be required by a newly developed application. Furthermore, removing the presentation logic leads to a simpler, easier to maintain application. In fact, the refactoring process should be seen as a business logic mining process.

Figure 4-19 illustrates what we would like to achieve in this stage.

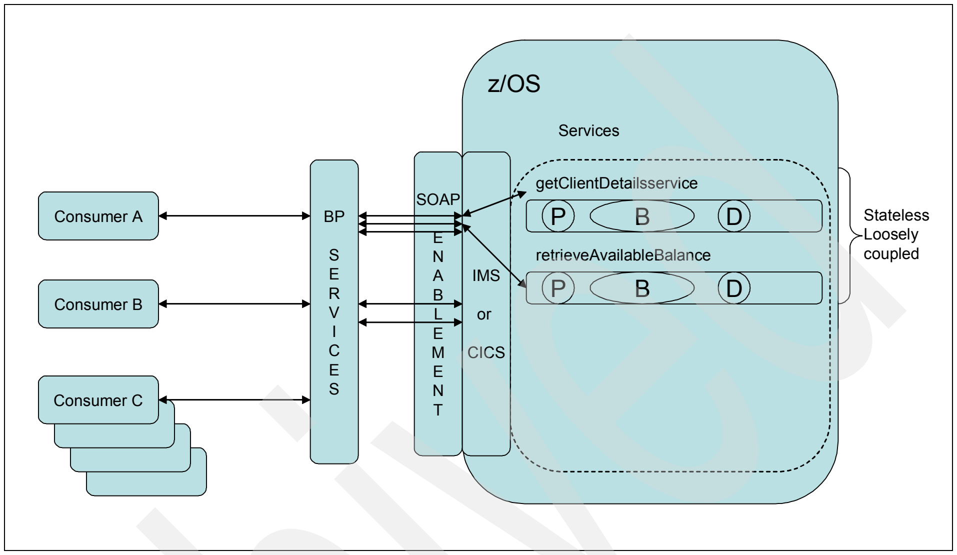


Figure 4-19   SOA-enabled business components in a 3-tier modular environment

As seen in "Common application patterns" on page 175, an application can take on many forms. Several of these patterns would assist in service enabling our new application; the choice of pattern influences the amount of effort involved in service enablement. Among the factors to consider are the following:

- Is the existing application modular or monolithic?
- Does the application contain presentation logic (3270, XML, proprietary, and so on)?
- Is business process choreography embedded within the application?
- What is the programming language (COBOL, PL/I, C++, C, Java, Assembler)?
- Does the application contain TP monitor-specific calls such as IMS ISRT or CICS SEND MAP?
- Are the 3270 programs still conversational?
- Are runtime-specific features used, for example CICS Temporary Storage Queues (TSQ)?
- To what extent are there already external channels linking into the legacy applications? Examples are clients connecting to IMS through IMS Connect

or OTMA directly, or clients connecting to CICS through Remote Procedure Calls (RPC), CICS TG, or even sockets. Is there a need to continue supporting “legacy” clients while performing the modernization and SOA enablement of the legacy programs?

Solutions for rejuvenation

In the following sections we discuss the solutions available for a rejuvenation approach. In all cases we assume that the solutions will be used in combination with the necessary refactoring to obtain the right level of service granularity and high reusability.

CICS Web Services

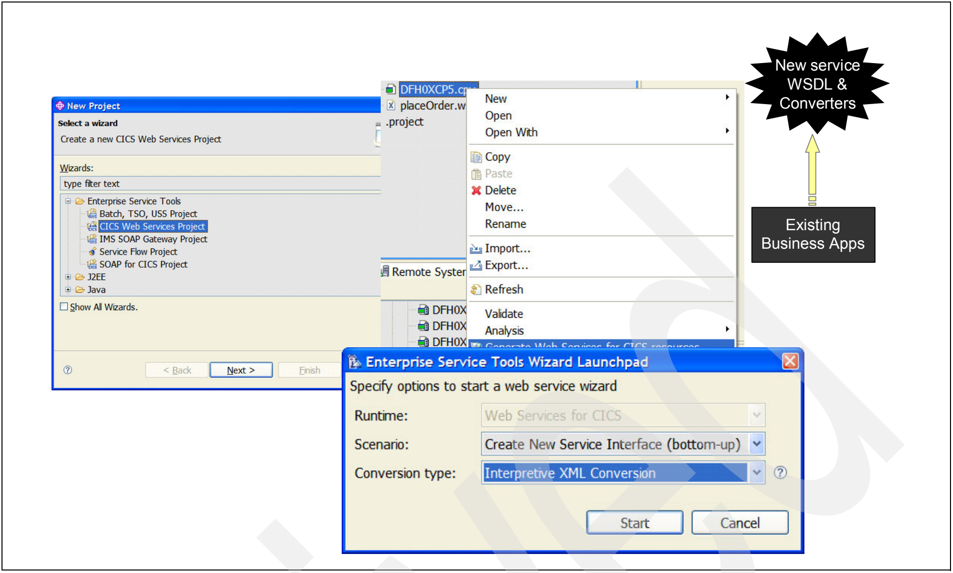
In this section we explore the tools and techniques that allow for existing CICS applications to be exposed as SOAP-based Web Services. The features described here are available with CICS TS Version 3.1 and later. We first look at the development process.

Development time tooling is provided within Rational Developer for z (RDz)[2]. Should you not wish to use RDz there are stand-alone batch utilities available as part of the Web Services Assistant utility. Using RDz is the more productive of the options. *XML Services for the Enterprise (XSE)* is the portion of RDz that deals with extracting metadata to provide runtime transformation between bytestream and XML, among other things. We only cover the RDz tooling in this section. For more details about the CICS Web Services Assistant batch utilities see the Redbooks publication *Implementing CICS Web Services*, SG24-7206.

CICS Web Services can be used in a “bottom-up,” “top-down,” or “meet-in-the-middle” approach. Because we are dealing exclusively with the rejuvenation of existing applications in this section we discuss the bottom-up approach only. This approach allows for an existing COMMAREA or language structure to be used and the resulting WSDL to be published and consumed.

See Figure 4-20 on page 205 for a snapshot of the functions in RDz.

[2] At the time of writing, RDz is available for limited use as a free download. See http://www.ibm.com/developerworks/downloads/r/rdz/learn.html for details.



Specify options to start a web service wizard

Runtime: Web Services for CICS

Scenario: Create New Service Interface (bottom-up)

Conversion type: Interpretive XML Conversion

Figure 4-20 XSE in RDz - Language structure to XML/XMI/WSDL

In reality, the language structure as detailed in the COMMAREA or copybook is often not ideal for direct exposure. One such example is the hiding of elements, whereby fields defined within a COMMAREA or language structure are required by the application but it makes no sense to expose these elements as part of the service interface. Consider the case of an application that provides customer address retrieval, in which case the COMMAREA expects an indicator function that can have one of two values:

1. retrievePostalAddress
2. retrieveResidentialAddress

In this case the option element within the COMMAREA should not be exposed as part of the SOAP service. In this example it is preferable to have two distinct SOAP operations. In this case the creation of a wrapper program to provide the fixed values of any elements not exposed is preferable. This wrapper program can also be valuable in providing any transformations necessary should the COMMAREA be too complex for the IDE to handle. The wrapper therefore allows the COMMAREA to be tailored into something that better suits SOAP exposure.

A word of caution: One of the inherent benefits of SOAP is the ability to describe the attributes of an element using an XSD either within a WSDL or referred to from within the WSDL. These attributes described in an XSD allow for validations to be created within the generated client proxy. You should therefore avoid changing an element’s basic attributes. For example, a numeric element should remain numeric and with the identical number of digits. The only exception to this rule usually relates to signs. From our experience we often find elements defined as signed when in truth a negative value is considered invalid. In this case it is recommended to adjust the COMMAREA/language structure to reflect what its true attributes are. This reduces the number of SOAP requests flowing from consumers that might fail basic level validation.

The next consideration is the transport protocol. CICS Web Services offers two options: HTTP/S and WebSphere MQ. This choice of transport protocol has an influence on the deployment infrastructure needed. Refer to Redbooks publication *Implementing CICS Web Services*, SG24-7206, for these deployment patterns.

This allows the fields defined within the language structure to now form part of the XSD internal to the WSDL. Also produced is a WSBIND file. This contains metadata that allows transformation to and from an XML document and a datastream matching the application's COMMAREA.

IMS SOAP Gateway

In this section we explore the tools and techniques that allow for existing IMS applications to be exposed as SOAP based Web Services. Some of the features described here are available with IMS v9 and others with v10.

Figure 4-21 on page 207 shows the various ways of accessing IMS programs as a service. The IMS SOAP Gateway is circled in red.

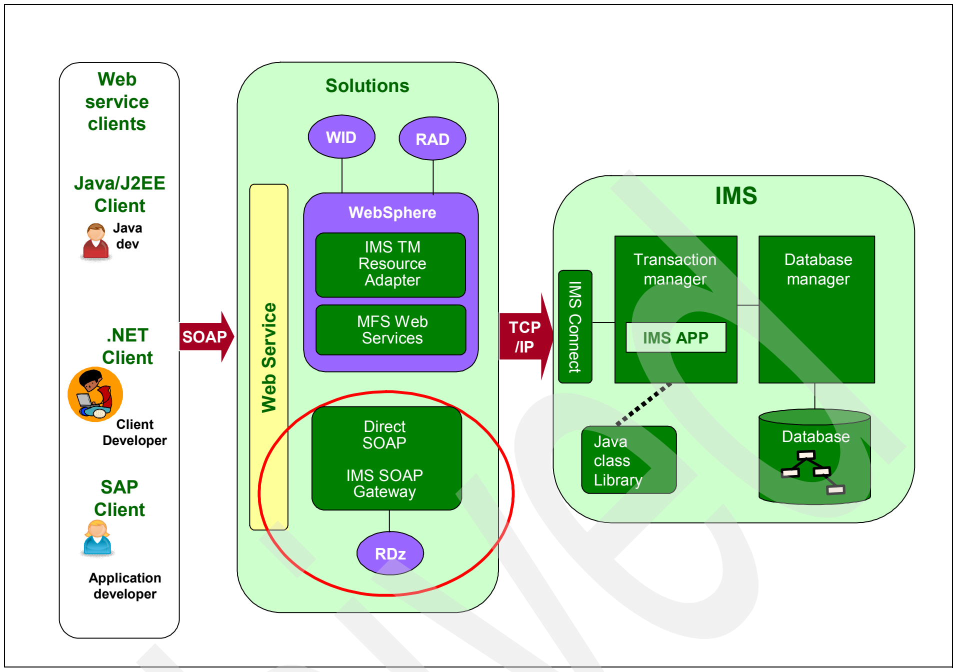
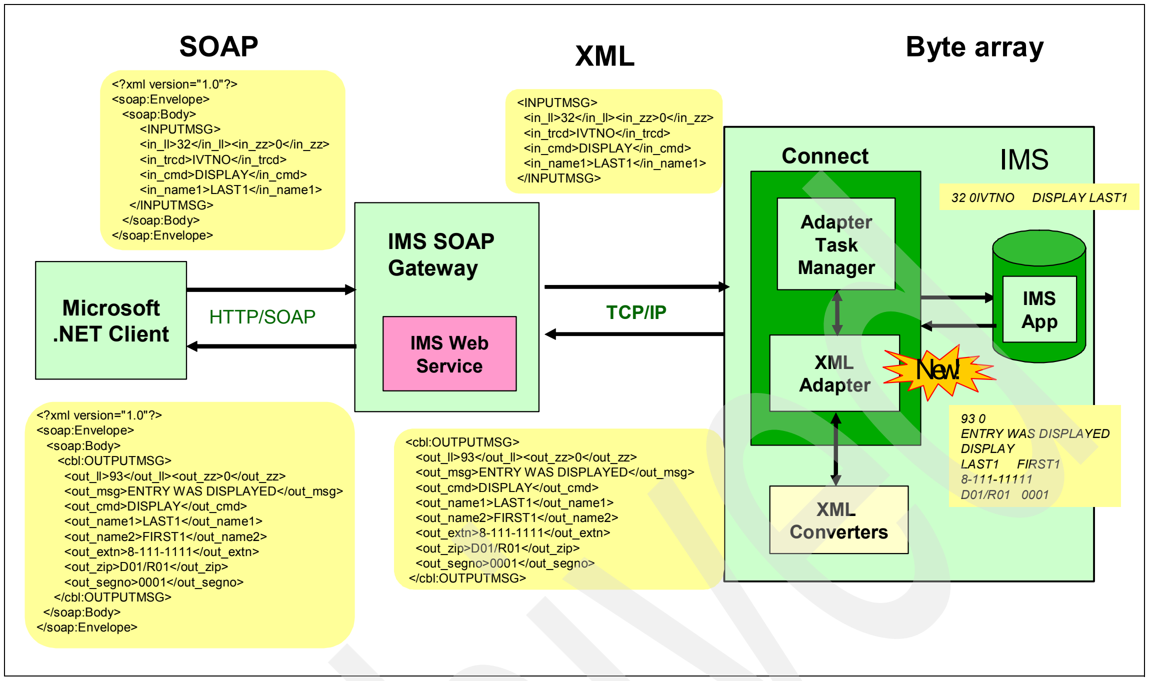


Figure 4-21 The IMS SOAP gateway architecture

The process of exposing an IMS application as a SOAP-based Web Service begins with Rational Developer for System z (RDz). The input to the process is the language structure (copybook) that maps both the input and output for this particular service. By identifying the input and output elements of the copybook RDz is able to generate a WSDL that describes this interface to a consumer. A further output of RDz is the XML converter files. These files contain executable code that provides transformation (XML de serialization and serialization) between an XML document and the datastream as is expected by the IMS application. These XML conversion drivers can then be implemented for use by the XML adapter function within IMS Connect. This transformation of the request and response ensures your existing application is relieved from any form of XML serialization/de serialization. Figure 4-22 on page 208 shows an example of this process.



```
<?xml version="1.0"?>
<soap:Envelope>
  <soap:Body>
     <INPUTMSG>
     <in_ll>32</in_ll><in_zz>0</in_zz>
     <in_trcd>IVTNO</in_trcd>
     <in_cmd>DISPLAY</in_cmd>
     <in_name1>LAST1</in_name1>
   </INPUTMSG>
  </soap:Body>
</soap:Envelope>
```

```
<INPUTMSG>
  <in_ll>32</in_ll><in_zz>0</in_zz>
  <in_trcd>IVTNO</in_trcd>
  <in_cmd>DISPLAY</in_cmd>
  <in_name1>LAST1</in_name1>
</INPUTMSG>
```

```
32 0IVTNO    DISPLAY LAST1
```

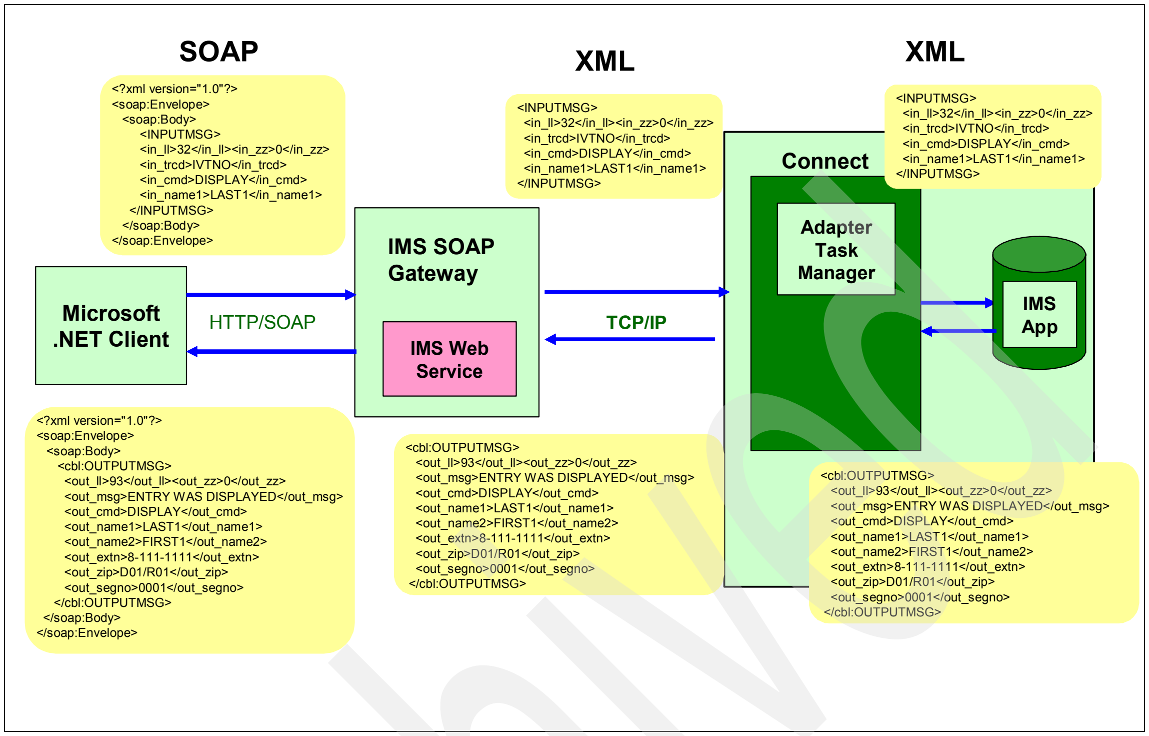
```
<?xml version="1.0"?>
<soap:Envelope>
  <soap:Body>
    <cbl:OUTPUTMSG>
      <out_ll>93</out_ll><out_zz>0</out_zz>
      <out_msg>ENTRY WAS DISPLAYED</out_msg>
      <out_cmd>DISPLAY</out_cmd>
      <out_name1>LAST1</out_name1>
      <out_name2>FIRST1</out_name2>
      <out_extn>8-111-1111</out_extn>
      <out_zip>D01/R01</out_zip>
      <out_segno>0001</out_segno>
    </cbl:OUTPUTMSG>
  </soap:Body>
</soap:Envelope>
```

```
<cbl:OUTPUTMSG>
  <out_ll>93</out_ll><out_zz>0</out_zz>
  <out_msg>ENTRY WAS DISPLAYED</out_msg>
  <out_cmd>DISPLAY</out_cmd>
  <out_name1>LAST1</out_name1>
  <out_name2>FIRST1</out_name2>
  <out_extn>8-111-1111</out_extn>
  <out_zip>D01/R01</out_zip>
  <out_segno>0001</out_segno>
</cbl:OUTPUTMSG>
```

```
93 0
ENTRY WAS DISPLAYED
DISPLAY
LAST1    FIRST1
8-111-11111
D01/R01   0001
```

Figure 4-22   Accessing an IMS Web service using the XML adapter in IMS Connect

A second option is to allow the XML payload to flow through IMS connect unhindered and have your application provide XML processing itself. This can be done with the assistance of the XML verbs available in Enterprise COBOL or via functionality offered by the XML Toolkit for z/OS. An example is shown in Figure 4-23 on page 209.



```
<?xml version="1.0"?>
<soap:Envelope>
  <soap:Body>
     <INPUTMSG>
       <in_ll>32</in_ll><in_zz>0</in_zz>
       <in_trcd>IVTNO</in_trcd>
       <in_cmd>DISPLAY</in_cmd>
       <in_name1>LAST1</in_name1>
     </INPUTMSG>
  </soap:Body>
</soap:Envelope>
```

```
<INPUTMSG>
  <in_ll>32</in_ll><in_zz>0</in_zz>
  <in_trcd>IVTNO</in_trcd>
  <in_cmd>DISPLAY</in_cmd>
  <in_name1>LAST1</in_name1>
</INPUTMSG>
```

```
<INPUTMSG>
  <in_ll>32</in_ll><in_zz>0</in_zz>
  <in_trcd>IVTNO</in_trcd>
  <in_cmd>DISPLAY</in_cmd>
  <in_name1>LAST1</in_name1>
</INPUTMSG>
```

```
<?xml version="1.0"?>
<soap:Envelope>
  <soap:Body>
    <cbl:OUTPUTMSG>
      <out_ll>93</out_ll><out_zz>0</out_zz>
      <out_msg>ENTRY WAS DISPLAYED</out_msg>
      <out_cmd>DISPLAY</out_cmd>
      <out_name1>LAST1</out_name1>
      <out_name2>FIRST1</out_name2>
      <out_extn>8-111-1111</out_extn>
      <out_zip>D01/R01</out_zip>
      <out_segno>0001</out_segno>
    </cbl:OUTPUTMSG>
  </soap:Body>
</soap:Envelope>
```

```
<cbl:OUTPUTMSG>
  <out_ll>93</out_ll><out_zz>0</out_zz>
  <out_msg>ENTRY WAS DISPLAYED</out_msg>
  <out_cmd>DISPLAY</out_cmd>
  <out_name1>LAST1</out_name1>
  <out_name2>FIRST1</out_name2>
  <out_extn>8-111-1111</out_extn>
  <out_zip>D01/R01</out_zip>
  <out_segno>0001</out_segno>
</cbl:OUTPUTMSG>
```

```
<cbl:OUTPUTMSG>
  <out_ll>93</out_ll><out_zz>0</out_zz>
  <out_msg>ENTRY WAS DISPLAYED</out_msg>
  <out_cmd>DISPLAY</out_cmd>
  <out_name1>LAST1</out_name1>
  <out_name2>FIRST1</out_name2>
  <out_extn>8-111-1111</out_extn>
  <out_zip>D01/R01</out_zip>
  <out_segno>0001</out_segno>
</cbl:OUTPUTMSG>
```

Figure 4-23 Accessing an IMS Web service without using the XML adapter in IMS Connect

Transformation of data elements from their format within an XML tag to one of the many formats available within z/OS languages is not a trivial task. For this reason we recommend, where possible, the XML conversion drivers as generated by RDz should be considered. By allowing XML de serialization to take place within IMS Connect there is also the added benefit of message size reduction prior to IMS processing.

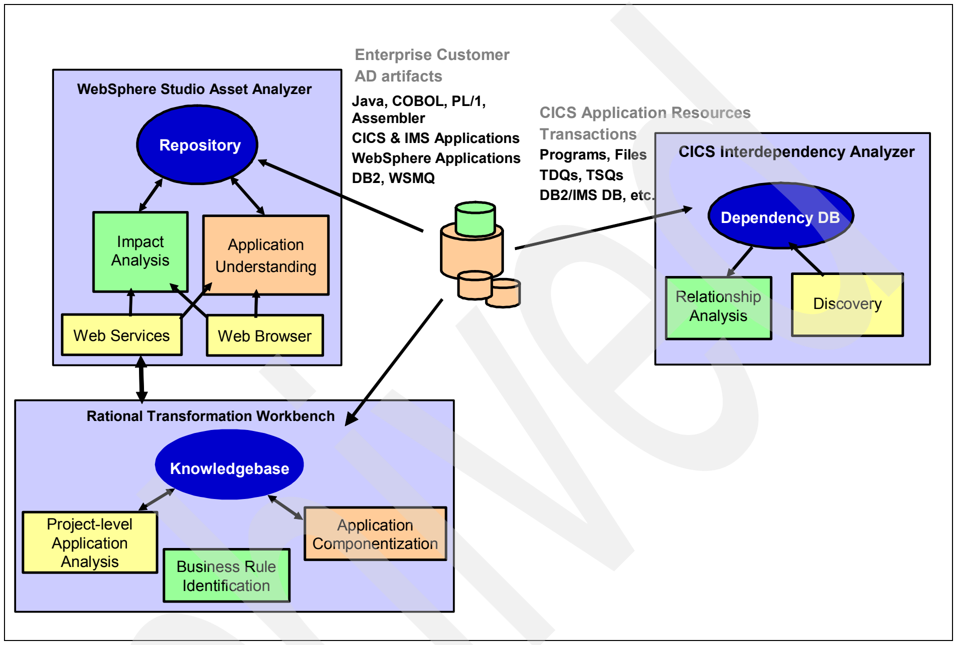
Tools for rejuvenation

A tool such as the *Rational Transformation Workbench (RTW)* is an ideal match for this rejuvenating process. Within RTW applications are imported into a repository called the *Enterprise Application Knowledgebase*. From here hidden knowledge about business rules and processes can be quickly analyzed and refactored. The knowledge base of application information provided by RTW can help reduce risks and costs incurred with enterprise application modernization initiatives such as moving to an SOA environment. RTW, together with tools such as *WebSphere Studio Asset analyzer (WSAA)*[3] and *CICS Interdependency*

[3] At the time of writing WebSphere Studio Asset Analyzer (WSAA) is in the process of being rebranded to Rational Asset Analyzer.

Analyzer (IA), is part of the *Discovery and Analysis* phase of the SOA life cycle model.

Figure 4-24 shows the three key offerings from IBM in Discovery and Analysis and their relationships.



Enterprise Customer
AD artifacts
Java, COBOL, PL/1, Assembler
CICS & IMS Applications
WebSphere Applications
DB2, WSMQ
CICS Application Resources
Transactions
Programs, Files
TDQs, TSQs
DB2/IMS DB, etc.
WebSphere Studio Asset Analyzer
Repository
Impact Analysis
Application Understanding
Web Services
Web Browser
CICS Interdependency Analyzer
Dependency DB
Relationship Analysis
Discovery
Rational Transformation Workbench
Knowledgebase
Project-level Application Analysis
Business Rule Identification
Application Componentization

Figure 4-24 RTW, WSAA and CICS AA

For more information about the capabilities of RTW see the following IBM publications:

- *IBM Asset Transformation Workbench Getting Started,* SC31-6877
- *IBM Asset Transformation Workbench Analyzing Programs,* SC31-6877

You can also get more information by visiting the RTW home page at
`http://www.ibm.com/software/awdtools/rtw/`

4.5.5 Solutions for replacing applications on System z

In this section we talk a bit about replacement of existing business application components as well as the development of new applications on System z.

Application replacement is a costly exercise. When compared to rejuvenation, a cost factor of five times more expensive is the typical benchmark. With such costs it is vital that the correct environment is architected to host such applications. Some of the key architectural decisions you must consider are the following:

- Language choice
- Development tooling
- Platform choice
- Information integration
- Application integration
- Application server/TP monitor

Language choice

The most important choice is the one between using a procedural and an Object Oriented (OO) language. In the procedural camp we have COBOL, PL/I, and C. For OO we have C++ and Java. Although Enterprise OO COBOL is available, we think it is best positioned as an integration mechanism between procedural COBOL and Java, as opposed to a mainstream development language.

A hybrid approach to this might be the use of *Enterprise Generation Language (EGL)*. EGL is a business-oriented language that offers the ability to generate COBOL or Java prior to implementation. Thus EGL falls into both the procedural and OO camps and it will therefore be discussed separately. Besides the technical aspects involved in language choice, something that might influence the language decision is the availability of programming skills.

Procedural languages typically do not enjoy a good reputation within IT. COBOL, for example, is often blamed for the inflexibility and cumbersome nature of existing applications. We believe, however, that this inflexibility stems from the nature of the application rather than the language itself. For example, a monolithic application does not allow for external process orchestration. Where we believe COBOL or PL1 might be considered as the language of choice is in cases where integration to existing fine grained or modular functionality is required. If this existing functionality falls outside the scope of your replacement project then COBOL or PL1 might be worth considering. This assumes you will retain CICS or IMS as your TP monitor of choice.

An Object Oriented approach using Java offers tremendous flexibility. The portable nature of Java and the reusability of Java classes are plus points. Using Java allows you to deploy the application to J2EE servers such as WebSphere Application Server for z/OS and WebSphere Application Server for Linux on System z, or even natively to CICS, IMS, or DB2.

Figure 4-25 provides the first part of a decision tree for such things as language choice, application and information integration, and skills availability. This decision tree does not cover all scenarios. Its purpose is to demonstrate some of the options available for developing new applications on System z or replacing existing applications with new applications on System z.

Note: This figure and Figure 4-26 on page 213 are two parts of the same diagram. Start reading from the left in the first figure, then at the end follow the A and B connectors to the second figure.

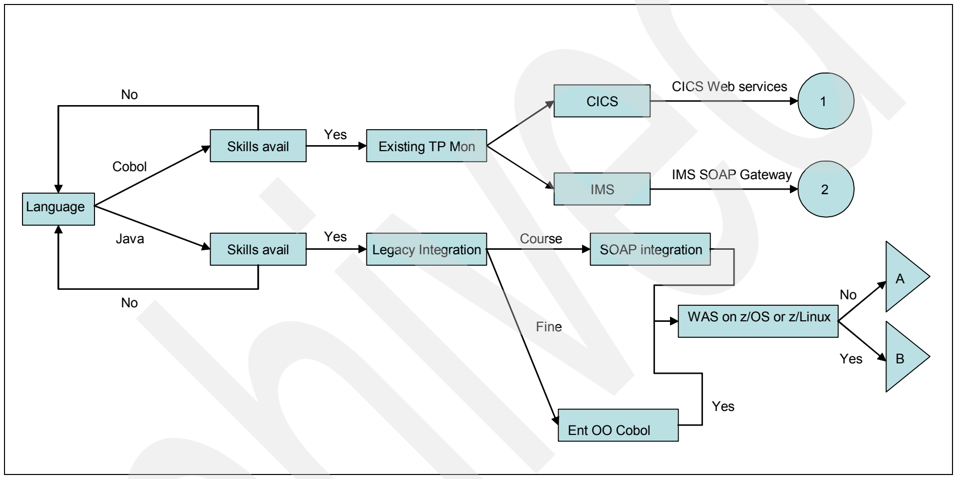


Figure 4-25 Part 1 of decision tree for new application development

Endpoints on the tree in Figure 4-25 result from making the following decisions:

1. TP monitor is CICS with 3 GL (COBOL, PL/I or C). Business Application services are modular, stateless, and loosely coupled. Business Application services are exposed via CICS Web Services.
2. TP Monitor is IMS with 3 GL (COBOL, PL/I or C). Business Application services are modular, stateless, and loosely coupled. Business Application services exposed via IMS SOAP Gateway.

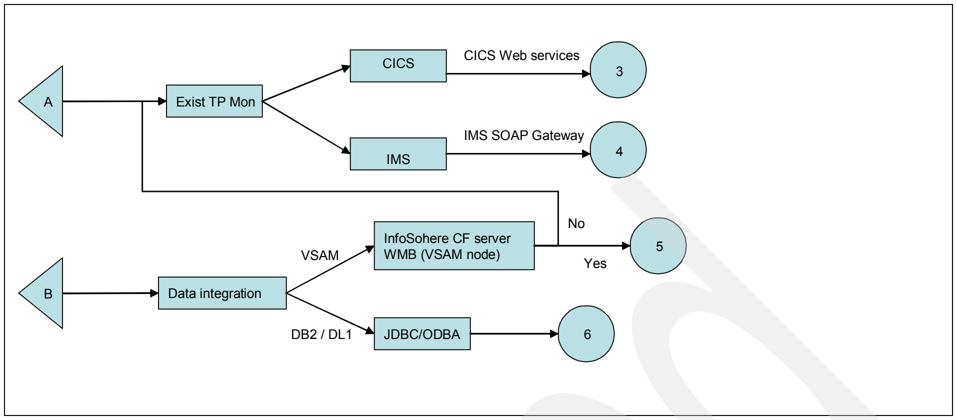


Figure 4-26 Part 2 of decision tree for new application development

Endpoints on the tree in Figure 4-26 result from making the following decisions:

3. TP Monitor is CICS with Java as primary language. Application and data integration to existing procedural components is via CICS APIs. Business Application services exposed via CICS Web Services.
4. TP Monitor is IMS with Java as primary language. Application and data integration to existing procedural components is via IMS APIs. IMS databases can be accessed using JDBC.
5. TP Monitor/Application server is WebSphere Application Server on z/OS or Linux on System z. Course grained application integration is via SOAP-based Web services. Access to VSAM data is via InfoSphere™ Classic Federation server for z/OS or WebSphere Message Broker via a VSAM node. If those products are not available, a CICS or IMS TP Monitor is used again for data access.
6. TP Monitor/Application server is WebSphere Application Server on z/OS or Linux on System z. Course grained application integration is via SOAP-based Web services. Access to DB2 and IMS DB is via JDBC.

Although the decision tree shown by no means provides for all application development scenarios on System z, it should provide some insight regarding some of the key architectural decisions required for developing applications on System z.

One of the scenarios not covered by the figures is the EGL language. EGL is a business-oriented procedural style language. It provides a level of abstraction between the application and the underlying middleware with all of its complexities. The goal of EGL is to increase productivity and quality of code by this abstraction layer. An added benefit of EGL is the ability to generate either COBOL or Java at any stage in the development process.

Developers using the Rational Developer for System z (RDz) and Rational Business Developer (RBD) are able to write and debug EGL.

Irrespective of the development language on System z, we believe that RDz offers substantial benefits to an organization. The productivity gains offered by a GUI Integrated Development Environment (IDE) over the traditional TSO/ISPF-based development environment are significant. For more information see the RDz home page at:

`http://www-306.ibm.com/software/awdtools/rdz/`

4.6 Financial implications

Cost has an impact on many architectural decisions. As discussed earlier in this chapter, budget constraints might influence whether you reface, rejuvenate, or replace an application.

The most important issue we believe is what return on investment is likely. An application refacing exercise might give fast returns and be relatively inexpensive but it does not add to flexibility and therefore reuse. An application might be replaced at high cost but will the solution provide the high returns required?

As a general rule, the most economical approach to SOA enablement on System z is to conduct the initial phase using a bottom-up mechanism. To reduce overhead an initial set of services should be built based on the following criteria:

- Easily identifiable
- Refactoring complexity (modular/Monolithic, embedded choreography, and so forth)
- Likelihood of reuse

These initial services are designed to bring high return for relatively little investment. Once SOA experience is gained within an organization, and possibly SOA benefits demonstrated across the corporation, a top-down or hybrid approach can be adopted. This approach is necessary in the longer term due to the nature of its more structured methodology and roles with the organization. However, this has an impact on cost.

4.7 Challenges and issues

Where applications on System z are well positioned for SOA is that a typical application often has a close match to the level of granularity required for a service. Other platforms and languages, which can be Object Oriented, might be very fine grained and require excessive amounts of orchestration to provide a service.

Where typical applications on System z are perhaps not as well suited to SOA are cases where applications have embedded presentation logic. The issue here is to choose the correct approach to maximize your SOA enablement. The biggest challenge usually facing a System z user relates to skills adoption. The average System z developer is usually not familiar with such things as TCP/IP, HTTP, XML, SOAP, and so forth. If a tiered approach is followed during the application rejuvenation process then an ideal business application will be presentation agnostic. In theory, this means a developer of a business application to be exposed as a SOAP-based service should not need to know XML, SOAP, HTTP, and so forth. However, in order to test and debug an application it is usually necessary to have a basic knowledge of such technologies. Fortunately, with testing tools such as the one provided with the Rational IDE, it is not necessary for a System z based developer to have knowledge of the platform the consumers of a service might finally deploy to.

5

Interaction services

While attention has been focused on services that can be extracted from traditional operational and line-of-business applications, other types of services can be just as easily incorporated into an SOA. Among the most important of these are presentation and collaborative services that are the basis for user interaction with the workflow and are key to the significant improvement in user productivity required today.

This chapter discusses the following topics with regard to Interaction services:

5.1 Definition, scope, and functions

In combination with presentation and collaborative services, SOA interaction services enable the creation of business processes that support and encourage teaming through local and remote collaboration. Such approaches can drive substantial innovation within the business by easing access to the knowledge that resides in users' heads and to information dispersed broadly across multiple data repositories.

5.1.1 Definition

Interaction services can be seen as the end user front-end to SOA. Interaction services are about the presentation logic of the business design or components that support the interaction between applications and end users. Also we recognize that interactions with the external world are not limited to just interactions with humans. In some cases, interaction logic orchestrates the interface to industrial robots, vehicles, sensors, RFID devices, environmental control systems, process control equipment, and so forth.

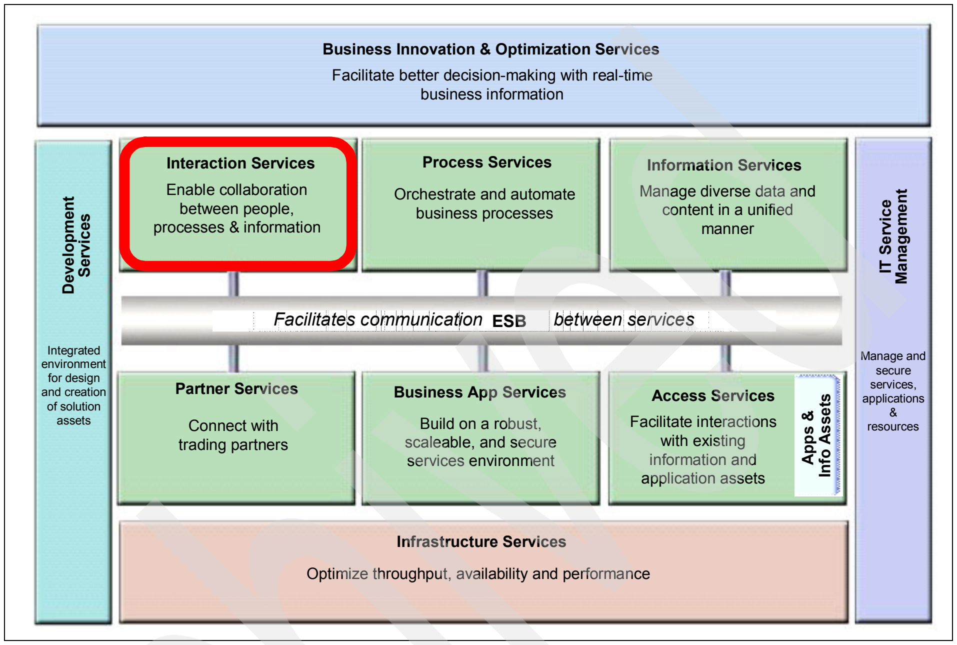
Regardless of to what or with whom your service is actually interfacing, every external interaction shares one thing in common: the ability to project a view of the information system tailored to the specific interaction fidelity, frequency of interaction, and presentation composition that best fits the needs of the end user or device. The most effective way of presenting information to one user to maximize their understanding might not be applicable to another user or situation. The most efficient way of gathering input to maximize the productivity of one user might not work for another user.

Interactions can be tailored to role-sensitive contexts by adjusting what is seen and the behavior presented to the external world based on who the user is, what role they are performing, and where they are in the world. Authentication, privilege-selection, and proximity might all be significant to what you can do and how.

Interaction services enable collaboration between end users of any kind – people, processes, and information.

5.1.2 Scope

Figure 5-1 highlights the scope of interaction services within the IBM SOA Reference Architecture.



Business Innovation & Optimization Services
Facilitate better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enable collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manage diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
Facilitates communication ESB between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitate interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimize throughput, availability and performance

Figure 5-1 IBM SOA Reference Architecture and interaction services

There is no business logic in interaction services because this is handled by the Business Application services; there is no data-related service in interaction services because this is covered by information services. Thus a Web Content Management solution is split between interaction services for the presentation and collaboration parts and information services for the content storage and management parts.

5.1.3 Functions and capabilities

Interaction services are services that give people or end users the ability to interact with the SOA infrastructure. Interaction services can be divided in groups of functions, specifically interface services, presentation services, collaboration services, content services, and integration services. These services are depicted in Figure 5-2 on page 220.

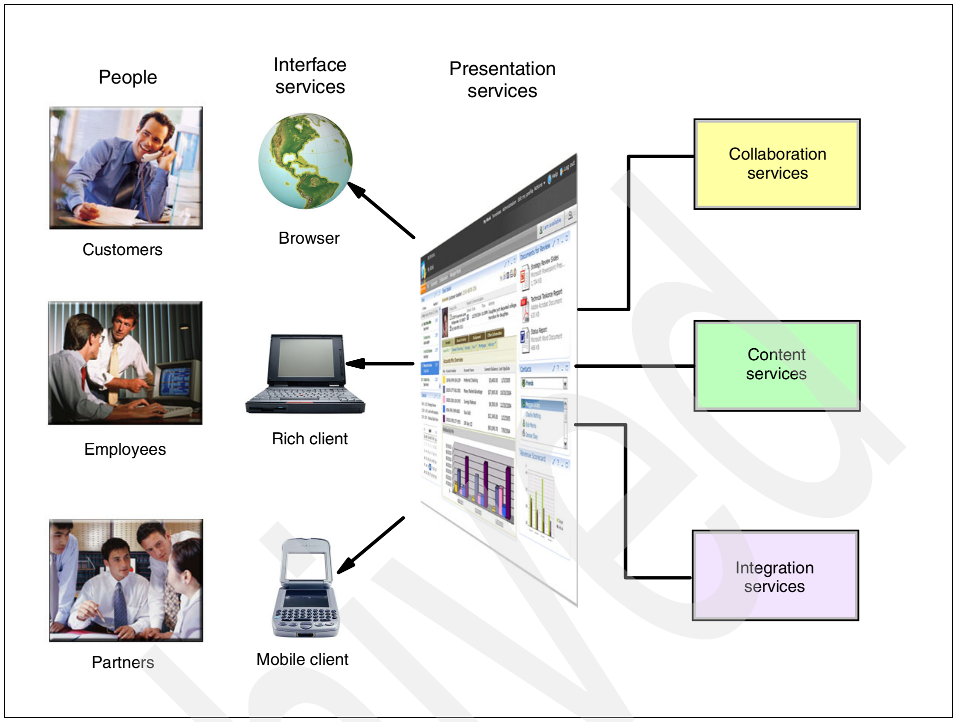


Figure 5-2 Interaction services functions

People or end users

People or end users are the human beings or terminals that interact with the SOA infrastructure. Using an interface, they can visualize information or they can trigger actions.

People can be customers, employers, partners, or suppliers — generally speaking, any individual who might have interactions with the infrastructure.

In some cases, interaction logic orchestrates the interface to industrial robots, vehicles, sensors, RFID devices, environmental control systems, process control equipment, and so forth.

Interface services

People or end users need an interface in order to interact with the infrastructure. Along with the evolution of the technology and the working environment, people or end users nowadays use an important variety of machines or devices to interact with systems.

The Internet has contributed to standardizing the user interface and the Internet browser is widely used as an interface for new applications. But Internet browsers have different capabilities depending on the software being used and the machine or the device they run on. Moreover, many services choose to use user interfaces other than Internet browsers, such as thick clients or proprietary device interfaces.

For example, the diversity of mobile devices continues to increase, as do their capabilities. Devices differ in their physical characteristics (screen size and keyboard), as well as in the markup languages supported by their browsers. Delivering a high quality user experience is increasingly dependent on device-aware tooling and runtime device recognition that can optimize delivery based on device-specific capabilities.

This is why Interface services are needed to customize the presentation content to the device hardware and software solution being used. This allows the enablement of pervasive technologies with simpler solutions.

Interface services can provide the following capabilities:

- Navigation, personalized experience, and multi-interface rendering
- User registration and profile management
- Security, access control, and single sign-on
- Aggregation of a broad mix of applications into a unified display
- A variety of deployment options across terminals, browsers, and devices
- Multi-channel access to presentation content
- Write once, render on multiple interfaces
- Optimization of the content based on the interface-specific capabilities.
- Support of standards

Presentation services

Presentation services build, aggregate, and organize the information coming from the business services in order to present it to the end user interface. They also provide the features to take actions with the business services. Presentation services get business information from collaboration services, content services, and integration services.

Presentation services improve people productivity by aggregating views that deliver information and interaction in the context of a business process.

Presentation services can contain a wide range of solutions — from a very simple Web presentation layer for one business application to an advanced portal with advanced personalization capabilities.

Presentation services can provide the following capabilities:

- Navigation and search
- Single point of entry for users
- Consistent look and feel
- Natural and adaptive user experience
- Intuitive user interface to promote access to targeted information
- Multi-channel delivery capability
- Advanced personalization and customization
- Information in the context of a business process
- Secure, modular, extensible, role-based workplace environments
- Single sign-on
- People awareness and personal content
- Composite display of multiple applications
- Support of standards
- Advanced user interface fly-out menus and page navigation, drag and drop
- Policy-based administration
- Aggregation of multiple applications' display and content in one view
- Aggregation of composite applications' presentation logic
- Customization of the computing desktop to match individual work patterns and roles

Collaboration services

Collaboration services is a set of services that allows business collaboration or communication among people. Collaboration services help people effectively share and manage information, make business decisions quickly, and streamline the way they work.

Collaboration services can contain a wide range of solutions — from a simple instant messaging application to an advanced workplace environment with e-mails, team spaces, and Web conferences.

Collaboration services can provide the following capabilities:

- E-mails with priority, security, and importance management

- Calendar features with invitation, meetings, and sharing capabilities
- Personal information management (PIM) functions, including personal directory and personal journal
- Discussion databases
- Teamroom applications, team spaces, and people finder
- Instant messaging with voice support
- Broadcasting tools
- Wiki and blog features
- Web conferencing
- Mobile or offline access
- Reference databases with basic workflow for document review
- Integrated administration and systems management tools

Powerful collaborative services allow users to quickly make business decisions with other employees, customers, and trading partners.

Content services

As unstructured content grows exponentially, there is a need to capture, store, manage, integrate, and deliver all forms of content across the enterprise. People need the right information, delivered in a timely manner, to make the best decisions.

Content services are aimed at providing an aggregated view of business information and providing ways to quickly organize and share content. Content services make business information or content available to and manageable from the Presentation services.

Content services can provide the following capabilities:

- Capture, storage, and management of content
- Imaging and digital asset management
- Aggregation and integration of content and business information
- End-to-end Web content management
- Content creation through WYSIWYG rich text editor or by importing from another application (such as word processing)
- Built-in content life cycle workflow with review and approval capabilities, automatic expiration of old content
- Versioning and rollback to previous content versions

- Federated search to find information throughout multiple repositories
- Dashboard views to ease the understanding of business metrics
- Personalization that delivers content that is relevant to the users and their roles and is dynamically delivered
- Publication to multiple Web sites as well as Workplaces

In the context of Interaction services, Content services is limited to creating, aggregating, interfacing, personalizing, and publishing the content for people. In a broader perspective, Content services also encompass advanced content management capabilities, which fall under the Information services chapter of this book.

Integration services

Integration services is a set of services which allows the integration of the presentation services with any other necessary service.

Integration services can provide the following capabilities:

- Support of standards (WSRP, Web Services, SOAP, and so forth)
- Support of a wide variety of communication protocols
- Integration with the Enterprise Service Bus
- Access to enterprise data
- Integration with process choreographer
- Presentation services to presentation services integration
- Support of remote presentation services
- Access to news feeds
- Integration with Enterprise Information Systems

5.2 Key architectural decisions with respect to System z

In this section, we present a checklist of key architectural decisions to be made with respect to System z and interaction services.

Before making any of these decisions, create a platform-agnostic list of functional and non-functional requirements.

- Decide and agree on a detailed and complete list of functional requirements.
- Decide if Interaction services are needed and to what extent.

- Decide and agree on a detailed and complete list of non-functional requirements (NFRs). All aspects of non-functional requirements must be analyzed. Section “Platform choice criteria” on page 232 can help in creating this list.

Once the detailed list of functional and non-functional requirements is created and agreed on, the following architecture decisions can be made:

- What security technical mechanisms or solutions satisfy the security requirements?
- What availability technical mechanisms or solutions satisfy the availability requirements?
- What performance technical mechanisms or solutions satisfy the performance requirements?
- What scalability technical mechanisms or solutions satisfy the scalability requirements?
- What transactionality technical mechanisms or solutions satisfy the transactionality requirements?
- What accounting solution satisfies the accounting requirements?
- Are the z/OS or the Linux on System z Qualities of Service required for some or all of these technical mechanisms or solutions?
- Should some interaction services be located on System z? If so, decide on the scope of interaction services functions or criteria for deploying these services on System z.
- Should the interaction services be federated between System z and distributed platforms? If so, decide whether a function can exist on both platforms and what the criteria are.
- What products satisfy the functional requirements?
- Should the products selected consist of a light solution with custom development needed, or should the products be a heavier, full-featured solution.
- What product features can be used to satisfy non-functional requirements?

5.3 Patterns

Patterns described in this section are runtime location patterns. They define where the interaction services will be located with respect to System z. Advantages and disadvantages of each pattern are explained.

We use the term “distributed” to refer to those platforms that are not System z based, such as x86- or UNIX-based platforms.

We discuss the following location patterns:

- Interaction services on distributed
- Interaction services on Linux on System z
- Interaction services on z/OS
- Interaction services federated on distributed and z/OS

5.3.1 Interaction services on distributed

In this pattern interaction services reside on the distributed platform while other services are either on distributed or System z, as shown on Figure 5-3 on page 227.

This is probably the first pattern that comes to mind when we think about where to deploy interaction services. This is because Web applications have entered the enterprise space with low-end non-critical applications. Moreover, the Internet and its unpredictable behavior (with slowdowns, cuts, failures) has made more and more end-users accustomed to bad Quality of Service. Hence it has become acceptable to have lower QoS for many applications, and their presentation logic was deployed on distributed platforms.

But as the enterprise is now transforming its core business assets into new models including SOA, the QoS of the interaction services comes into play. It might still be acceptable to have lower Quality of Service for the presentation services of some applications, but it is not anymore for some more critical applications. This is particularly true for applications whose presentation logic used to run in Enterprise Information Systems such as CICS or IMS, and whose users have been accustomed to high availability, high security, high resilience, and so on. Moreover, interaction services include a wide variety of services (interface, presentation, collaboration, content, integration), which might each need some different levels of QoS.

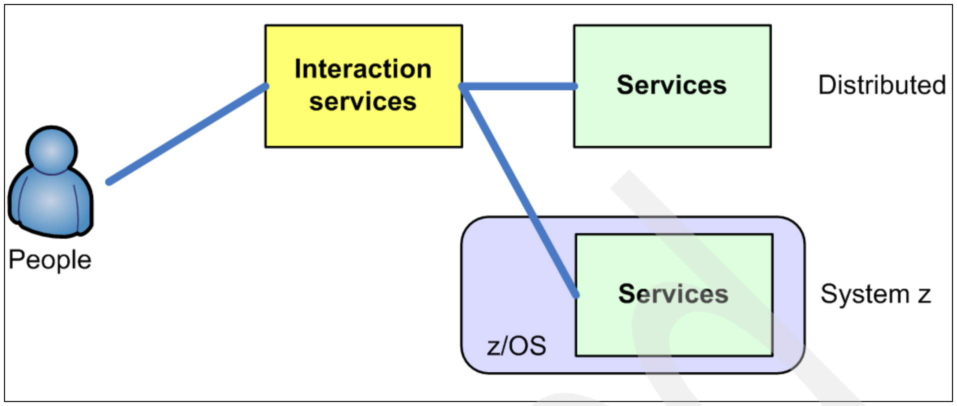


Figure 5-3 Pattern with Interaction services on distributed

The advantage of having interaction services deployed on the distributed platform is that mainframe capacity can be reserved for types of application logic that require a higher Quality of Service.

One disadvantage of having interaction services deployed on the distributed platform is that the interaction services do not benefit from the resilience provided by System z. Moreover, there is no possibility to achieve high QoS for critical interaction services. Even if most interaction services do not require the highest QoS, the few interaction services that do need it cannot afford not to have it. Some examples of interaction services that require the highest QoS are those where:

- Encryption keys must be stored in a highly available secured cryptographic hardware.
- A very scalable infrastructure is needed, with the capability to start software components dynamically depending on the business performance goals.
- Very high throughput with a repository is needed to access all data or content.

More Quality of Service criteria that can help in the platform choice decision for interaction services are discussed in section 5.4, “Platform choice criteria” on page 232.

5.3.2 Interaction services on Linux on System z

In this pattern interaction services reside on Linux on System z while other services are either on distributed or System z, as depicted on Figure 5-4.

Linux on System z is well known for aggressively exploiting virtualization with z/VM, for addressing the growing software and data center costs, for simplifying environments, and for its ability to significantly improve the provisioning process.

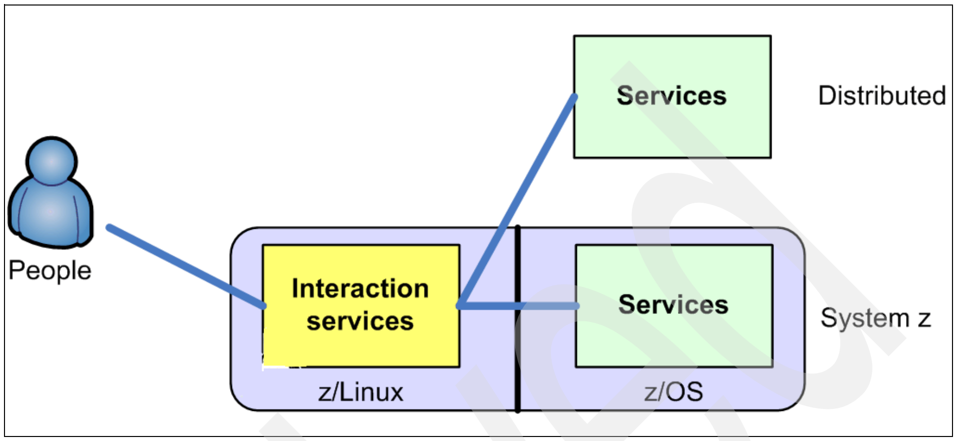


Figure 5-4 Pattern with interaction services on Linux on System z

One advantage of having interaction services deployed on Linux on System z rather than on distributed is to benefit from System z hardware resilience. This pattern also has the advantage of consolidation of workloads on one System z machine. The consolidation of different workloads, along with the virtualization of hardware resources, allows for sharing and better utilization of the hardware resources in order to reduce total hardware capacity. Another advantage is the proximity to services deployed on the z/OS platform. For example, having interactions services on Linux on System z and other services on z/OS on the same hardware allows for a higher level of security with no network communication, and higher performance with memory-to-memory communication (HiperSockets). Also, having interaction services on Linux on System z minimizes hardware complexity because using the same server infrastructure as z/OS reduces connections to storage and to external systems.

One disadvantage of having interaction services deployed on Linux on System z platforms is that there is no possibility to benefit from the z/OS Quality of Service features (such as enhanced security, transactionality management, and scalability) for critical interaction services. Even if most interaction services do not require the highest QoS, the few interactions services which might need it cannot benefit from it.

More Quality of Service criteria that can help in the platform choice decision for Interaction services are discussed in section 5.4, “Platform choice criteria” on page 232.

5.3.3 Interaction services on z/OS

In this pattern interaction services reside on z/OS while other services are either on Distributed or System z, as shown on Figure 5-5.

Having the Interaction services deployed on z/OS allows us to leverage the z/OS QoS features as well as the specific Interaction services software capabilities that are unique to this platform.

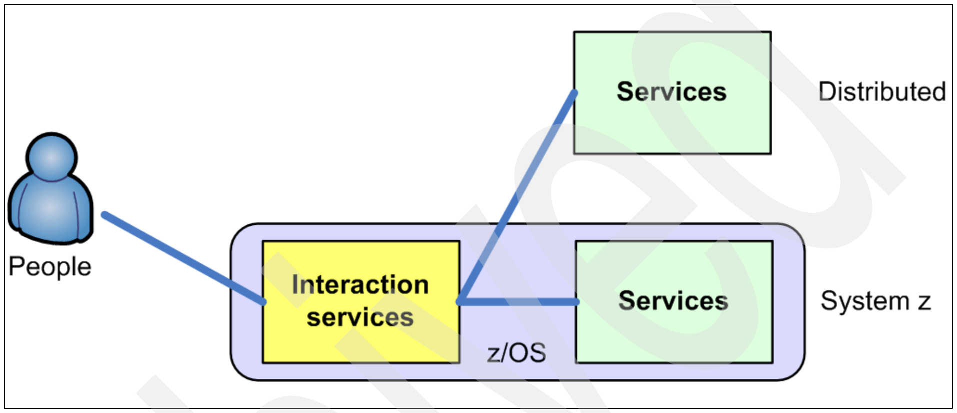


Figure 5-5 Pattern with interaction services on z/OS

One advantage of having interaction services deployed on the z/OS platforms is to benefit from the System z hardware resilience. Another advantage is that interaction services benefit from the z/OS highest Quality of Service for critical interaction services. For example, some interaction services might require a very scalable infrastructure with the capability to start software components dynamically depending on the business performance goals. There are also cases where the interaction services require very high throughput to access all their data or content.

One disadvantage of this pattern is that some of the interaction services might not need the z/OS highest QoS and might fit better on a distributed platform.

More Quality of Service criteria that can help in the platform choice decision for interaction services are discussed in section 5.4, “Platform choice criteria” on page 232.

5.3.4 Interaction services federated on distributed and z/OS

In this pattern interaction services reside on z/OS and on the distributed platform and other services are either on distributed or System z, as shown in Figure 5-4.

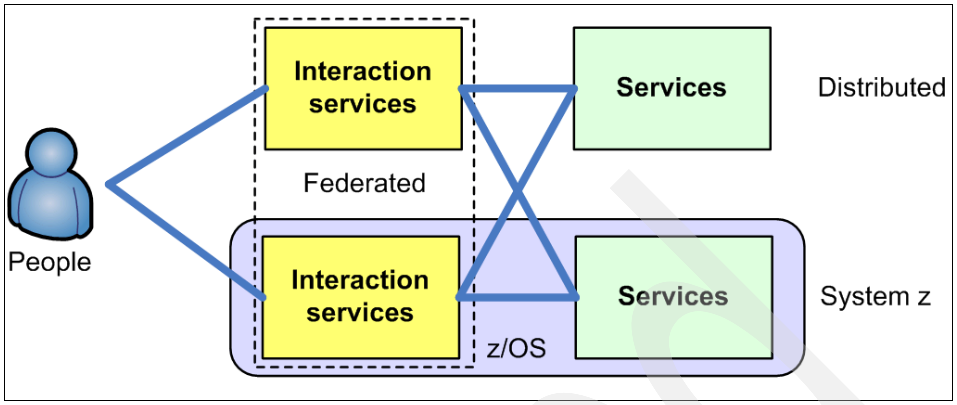


Figure 5-6 Pattern with Interaction services federated on distributed and z/OS

The interaction services are federated so that this is transparent to people or end users.

The federation can have multiple shapes in this pattern. Some interaction services functions can be deployed on distributed while other functions can be deployed on z/OS. Figure 5-7 on page 230 shows this pattern with a group of interaction services functions deployed on distributed (F1) and another group of interaction services functions deployed on System z (F2). This pattern is relevant when functions have different QoS requirements. For example, there can be presentation services such as the presentation of a core banking application deployed on z/OS and collaboration services such as instant messaging deployed on distributed. The core banking application might require high security while the instant messaging solution does not need high levels of availability.

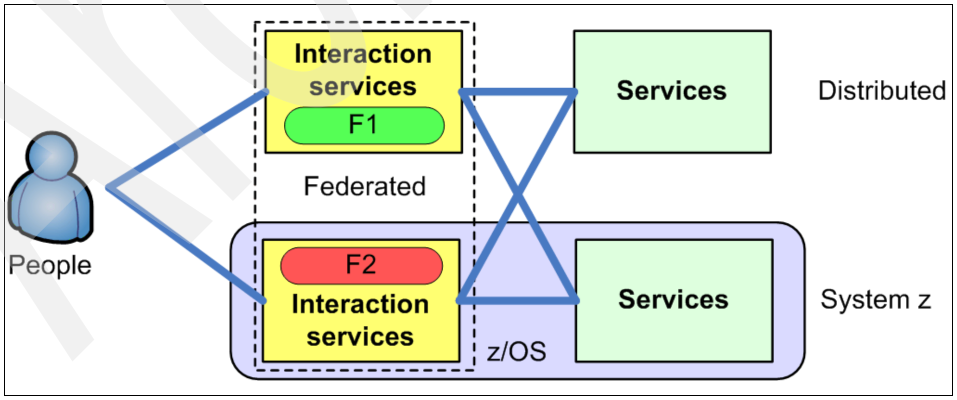


Figure 5-7 Pattern with interaction services federated and separated functions

Another shape for this pattern could have some interaction services functions deployed on both distributed and z/OS, and some subsets are deployed on one or the other platform. Figure 5-8 shows this pattern with a group of interaction services functions deployed on distributed and z/OS (F1d and F1z) and another group of interaction services functions deployed on distributed and z/OS (F2d and F2z). This pattern is relevant when the same function can have different Quality of Service requirements depending on other criteria such as group of users or time. For example, there can be collaboration teamroom services deployed on z/OS for critical business content and users and collaboration teamroom services deployed on distributed for less important content and users.

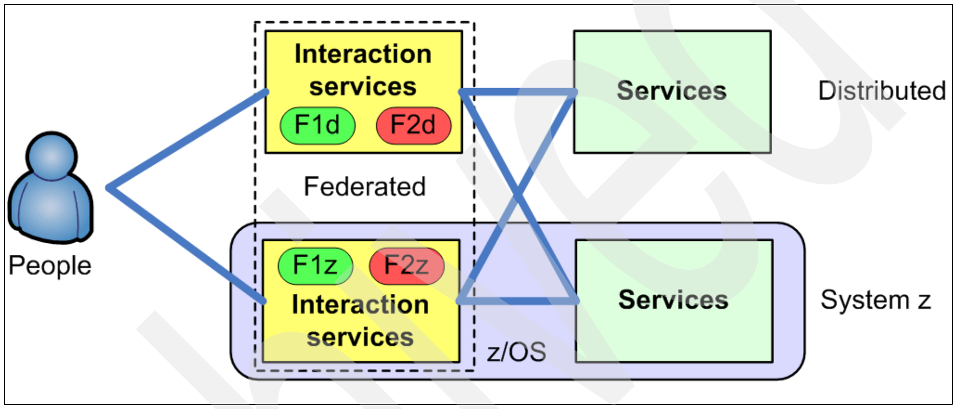


Figure 5-8 Pattern with interaction services federated and same functions split

One advantage of having interaction services deployed on the z/OS platforms is to benefit from the System z hardware resilience. Another advantage is that interaction services benefit from the z/OS highest QoS for critical interaction services.

One disadvantage of having interaction services deployed on the z/OS and the distributed platform is the added complexity to design, implement, and operate the federation of interaction services. If different interaction functions are deployed on the z/OS and distributed platforms, then the additional complexity is limited because it is very similar to having separated standalone solutions. If the same interaction functions are deployed on both the z/OS and distributed platforms but with different sub-sets, then either the solution can handle it by itself so that it is transparent to the end users and administrators, or the solution cannot handle it and you will need to put in place specific procedures and tools to handle this situation.

5.4 Platform choice criteria

The choice of a platform includes the selection of hardware and software that satisfy the functional and the non-functional requirements of a solution design.

The beginning of this section focuses on the capabilities and functions that interaction services can possess. These are the functional requirements for the interaction services solution. In the solution design process, it is also necessary to make a list of the non functional requirements for the interaction services solution.

Unfortunately, the non-functional requirements of a solution also tend to be treated as minor considerations because they do not add any new or improved functionality. Thus, they typically do not receive the proper attention from executives, the project manager, or even the technical team. However, a project must address issues such as security, availability, and performance in all phases of a solution design life cycle to be successful.

This section describes the criteria that need to be considered when choosing a platform during the Interaction services solution design.

5.4.1 Satisfying functional requirements

It is common but incomplete to see functional requirements driving the choice of software. Then the hardware is chosen depending on other considerations such as non-functional requirements. One flaw in this approach is that software capabilities are different depending on hardware and operating systems. A complete hardware and software solution should be planned at the same time to ensure the very best platform selection is made. Therefore, not only functional requirements, but also non-functional requirements must be considered at the same time and with the same level of importance when deciding which platform to choose.

Functions and capabilities of the interaction services are described in 5.1.3, “Functions and capabilities” on page 219. These can be used as a starting point for making a list of the functional requirements for the Interaction services solution.

Due to the variety of functions and capabilities that are part of Interaction services, it makes sense to group subsets of functions and to look for a hardware and software solution for each subset. In addition, if non-functional requirements differ a lot for one function with several different uses, then it make sense to look for a hardware and software solution for the non-functional requirements specific to each use.

For example, Interaction services contains functions dealing with Presentation services and others dealing with Collaboration services. Products all have different functions and cover different scopes, so it is quite difficult to compare product features and then see if they satisfy the functional requirements. This is the reason why architects should start with a clear product-agnostic list of functional and non-functional requirements and then have everybody agree on this list of generic requirements.

Different scopes of functional requirements might end up in completely different solutions, so it is important to have an exhaustive list of requirements with their scope, and everything *not* mentioned in the list is not necessary. Also plan for requirements that might come in the near future, and evolving requirements, so that the chosen solution will be able to adapt without changing the complete infrastructure. For Interaction services one scope could be instant messaging, another could be e-mail. Another scope could be a Web front end or a portal, and there are many others.

Once the list of functional requirements and non-functional requirements with their scope is complete and agreed on, architects can look at solutions and evaluate how these solutions satisfy all requirements.

For example, if there are requirements for simple presentation services to some existing business services, a lightweight Web application server solution might be sufficient. If there are requirements for complex navigation and personalization capabilities, a more advanced Portal solution might be necessary.

5.4.2 Security

Security is usually part of the non-functional requirements. Again, non-functional requirements should be listed at the same time and with the same importance as functional requirements. Once agreed upon, functional and non-functional requirements can be set against software and hardware solutions' functions and capabilities.

Because Interaction services are most often the front end to the SOA infrastructure, security plays an important role in making sure that common security disciplines are applied, such as identification, authentication, authorization, integrity, confidentiality.

Here are some security requirements that apply to the Interaction services and that are pertinent to the platform choice:

- **Centralized security management**
  There are many functions and capabilities within Interaction services. Along

with the growing number of Presentation or Collaboration servers, such as Web applications or e-mail servers, there is a need to centralize the security management to simplify administration, enforce consistent policies, and ensure that robust security mechanisms are applied.

- **External authentication in a DMZ**
  When dealing with the unsafe Internet world or even within a corporate intranet network, it is common to build *DeMilitarized Zones (DMZ)* made of network firewalls. Then security policies often stipulate to authenticate end users as soon as possible, which can be in a DMZ. *Reverse Proxy Security Servers (RPSS)* are used for this purpose. Authentication does not occur in a server running Presentation services; instead, it occurs outside. This is called external authentication and Presentation services must support it.
- **External authorization in the security manager**
  The authorization decision to give access to a resource can be made in the component where the resource is or outside in the security manager. Because of centralized security requirements for Interaction services and particularly for Presentation services, it is more and more common to require external authorization in the security manager.
- **Single sign-on**
  For end user convenience, it is preferable to have the end user logging in once at the beginning and then keeping the user logged in for any application he uses. This is a common requirement for Presentation services.
- **Identity propagation**
  As tight security becomes more important and as more and more regulatory and compliance requirements appear, there is a need to track and audit what users are doing end-to-end. For this purpose the end user identity must be propagated from the front-end Presentation services to any back-end service and through any intermediary service such as an ESB.
- **Support of standards**
  Standardization offers lots of advantages such as new integration capabilities and interoperability across solutions. There are many standards for Interaction services and security, such as WS-Security, J2EE security, SAML, and so forth. Choosing a solution that complies with those standards allows greater flexibility.
- **Minimal network communication**
  Depending on how critical the Interaction service is, there might be a need to minimize network communication in order to achieve higher levels of security. For example, this can be true for core banking applications accessed by superusers with the authority to invest large amounts of money.
- **Highly secured truststore for keys**
  Depending on how critical the Interaction service is, there can be a need to secure the transport channel with encryption and maybe client certificate

authentication. Encryption requires a *truststore* where keys are stored. This is a critical component which must be highly secured in order to preserve the integrity and the security level of the solution. A highly secured truststore can be managed by keys stored in a hardware component, for instance.

- **High security isolation between environments**
  Depending on how critical the Interaction service is, there can be a need to have a secured isolation between environments so that one cannot compromise the other. Because there is more and more consolidation with sharing of hardware resources, secured isolation (between LPARs, for instance) becomes more important.
- **Auditing of all security events**
  A very important feature of a centralized authentication and access control mechanism is the ability to record and analyze security information. The audit data is essential for ensuring that the customer's installation security policy is being followed. Thus any Interaction service must be able to build security auditing data and this data must be centralized for further analysis.

5.4.3 Availability

It is important that end users be able to conduct business and access the proper business services at any time during their working hours. Because Interaction services is most often the front end to the complete SOA infrastructure, some Interaction services such as Presentation services to business services need high availability in order to keep the business activity running.

Here are some availability requirements that apply to the Interaction services and that are pertinent to the platform choice:

- **Clustering of software instances**
  Redundancy of software is a common requirement in order to improve availability. Redundancy can take the shape of clusters of servers.
- **High availability of the hardware**
  Redundancy of hardware is also a common requirement in order to guarantee high availability.
- **Workload balancing mechanisms based on performance goals**
  High availability and clusters of software instances imply the use of load balancing solutions. Some Presentation services require some consistent response times or must satisfy some Service Level Agreement (SLA). For example, a bank branch office might have a high number of complex transactions to complete within a day. If the complex transaction response time is too high, then the branch office is not able to execute all required complex transactions within a day. Depending on resource utilization, some smart load balancers can choose where to load balance the workload to

achieve the performance goals. This is sometimes called dynamic load balancing.

- **Robust data synchronization mechanisms**
  When multiple instances of software are involved and when stateful applications are concerned, there is a need to have a synchronization mechanism. For critical applications this synchronization mechanism must be robust in order to recover well from any failure.
- **High availability and workload sharing of the datastore**
  Interaction services might make heavy use of a datastore. For example, a Presentation service Portal solution might access a lot of customization data for each user or a lot of intranet content; a Web content management solution might access a lot of data to store documents. Then the datastore needs to be highly available and with workload sharing for better resource utilization.
- **Seamless hardware and software upgrades**
  Because of consolidation of multiple kinds of workloads on the same hardware and sometimes on the same software, it is common to have a requirement for non disruptive hardware or software maintenance.
- **Availability manager to restart failed instance**
  For good Quality of Service, an availability manager can monitor running instances and try to restart any failed instance. This is a good way to prevent any human error, for instance.

5.4.4 Performance

Performance requirements depend on the Interaction service functions and usage. A Collaboration teamroom used from time to time for not so important documents might have no performance requirement. But core business applications that are critical for running the business every day have performance requirements. For example, a bank branch office might have some high performance requirements for applications used by customer-facing employees all day long. The bank cannot tolerate to loose or to dissatisfy customers because of bad response times.

Here are some performance requirements that apply to the Interaction services and that are pertinent to the platform choice:

- **Workload manager based on performance goals**
  For good response times for critical applications, there is a need for a workload manager that is able to monitor the performance goals and make the appropriate decisions when needed. Taking action means prioritizing the work by giving higher priority and assigning more resources to critical applications. This is even more critical with consolidated environments where

multiple types of workload run on the same hardware. The workload manager should also be able to tell whether the performance goals are reached.

- **Assign specific performance goals for transactions**
  Not all Interaction services require the same allocation of hardware resources because they do not all have the same performance requirements. There is a need to differentiate the services based on performance requirements and to manage them differently.
- **High I/O bandwidth to access content or personalization data**
  For Interaction services that make heavy use of content or personalization data, it is important to provide high I/O bandwidth to the datastore and to the disks.
- **Proximity to data and business services**
  For Interaction services accessing business services or data services with very high performance goals, proximity or even co-location with those business services or data services can be a requirement.
- **Encryption performance**
  For Interaction services with a high level of security, such as an online store with purchase capabilities, encrypting the transport channel with SSL/TLS is common. Then if there are high performance requirements or if the workload is important, there is a need for hardware cryptography for the highest performance.

5.4.5 Scalability

With more and more services facing the Internet, workloads become increasingly unpredictable and infrastructures must be flexible and scalable in order to absorb peaks and provide adequate performance.

A scalability requirement that applies to the Interaction services and that is pertinent to the platform choice decision is *dynamic scalability*. Predictable workload changes can be handled by humans or automated. Unpredictable workload or very changing workloads require a scalable infrastructure. The infrastructure needs to evaluate the performance of the system and evaluate the need to start or stop new resources. Some Interaction services require this flexibility because user activity fluctuates over time.

5.4.6 Transactionality

Some specific Interaction services might have particular transactionality requirements.

Transactionality requirements that apply to the Interaction services and that are pertinent to the platform choice are *Two-Phase Commit with distributed services* and *Two-Phase Commit with Enterprise Information Systems services*.

5.4.7 Accounting

A data center has to keep track of resources to determine its expenses so that they can be accounted to the business units. This is particularly relevant for the Interaction services that are the front end to SOA, where one needs to know who used which resources for accounting and billing purposes.

An accounting requirement that applies to the Interaction services and that is pertinent to the platform choice is *resource usage tracking.* End users access the Interaction services, which then access data, processes, or business services. Thus the Interaction services are the first services to know which user is using which service. Interaction services need to keep track of services being called and resource consumption.

5.5 Presentation solutions

In this section we discuss Presentation solutions appropriate for the System z platform and also how to integrate with System z assets.

5.5.1 Choosing a new simple Web application server on System z

As a reminder, Presentation services can provide capabilities such as navigation, search, single point of entry for users, consistent look and feel, single sign-on and many other capabilities. Depending on needs and requirements, it might be sufficient to use a J2EE application server as a starting infrastructure for the Presentation services. It is possible to develop all the Presentation services on a J2EE standard application server.

J2EE is a platform-independent, Java-centric environment for developing, building, and deploying Web-based enterprise applications on line. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tier, Web-based applications.

This solution is appropriate when only a small subset of the Presentation services capabilities are required. For example, a J2EE application server can easily provide a consistent look and feel and a single entry point to some business services. But it is more difficult to develop, manage, and maintain the composite display of multiple applications or people awareness.

The IBM products that satisfy these requirements are part of the WebSphere Application Server family. WebSphere Application Server is the foundation of the IBM WebSphere software platform and a key building block for a service-oriented architecture.

Running on System z, different flavors of this application server exist which differ in terms of capabilities and Qualities of Service:

- *WebSphere Application Server* which runs on Linux on System z
- *WebSphere Application Server for z/OS*
- *WebSphere Extended Deployment* which runs on Linux on System z
- *WebSphere Extended Deployment for z/OS*

WebSphere Extended Deployment extends the Quality of Service of WebSphere Application Server with additional capabilities such as dynamic application placement, health management, workload traffic shaping, scalable data fabrics, and so forth.

WebSphere Application Server running on Linux on System z passively benefits from the System z hardware, especially virtualization features. Solutions requiring speedy deployment with less stringent QoS or integration requirements are especially suitable to WebSphere for Linux on System z.

WebSphere Application Server for z/OS actively exploits the System z hardware and z/OS. Solutions requiring high QoS and significant integration with CICS, IMS, or DB2 will be best served by WebSphere Application Server for z/OS.

Figure 5-9 shows an example of a WebSphere Application Server for z/OS deployment in a cluster configuration. It shows the multiple Servant (Server) regions for dynamic scalability.

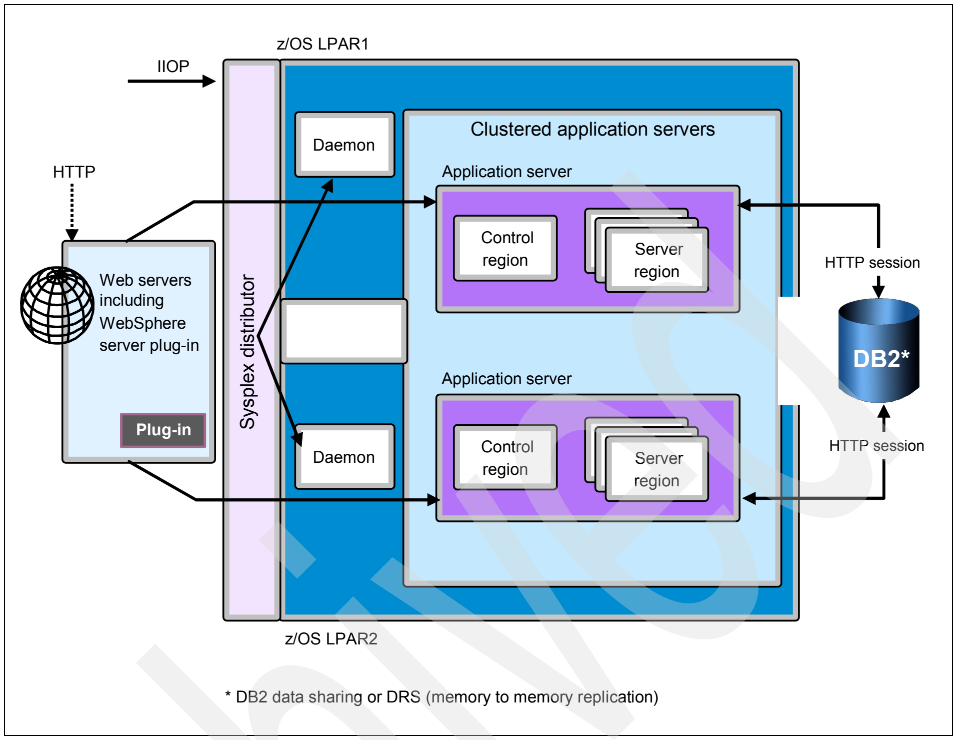


Figure 5-9 WebSphere Application Server for z/OS in a cluster configuration

Both the Linux on System z and z/OS versions of WebSphere Application Server adhere to the WebSphere development model and tools. Both adhere to the WebSphere systems management and administration model and tools. Both provide a unique value-add in the deployment of WebSphere applications. The z/OS value proposition is to provide the highest possible Quality of Service in an efficient, cost-effective manner. The Linux value proposition is to simplify and optimize existing infrastructure for end-to-end WebSphere applications with the goal of reducing costs and complexities.

For integration in the larger SOA environment or more specifically with an ESB, WebSphere Application Server supports a large set of connectors and adapters that handle synchronous and asynchronous communication. Most of the connectors rely on connection pooling for high availability and high performance (Datasources, Connection Factories). WebSphere is fully compliant with the *Java Connector Architecture (JCA)*. WebSphere Application Server for z/OS is particularly well suited for integrating with Enterprise Information System with specific QoS. For example, the use of local connectors to CICS, IMS, or DB2 allows for propagation of security identity and provides 2-Phase Commit support to run global transactions.

WebSphere Application Server is the base for WebSphere Portal, so most of the discussion about how to answer some of the Presentation services platform choice criteria with WebSphere Portal also applies to WebSphere Application Server. Section 5.5.2, “Choosing a new advanced Portal on System z” on page 244 provides further discussion about how to satisfy some of the Presentation services requirements.

To help with the platform choice, Table 5-1 on page 241 through Table 5-5 on page 243 highlight the most significant Quality of Service features provided by the WebSphere Application Server products on both Linux on System z and z/OS.

Table 5-1 lists features highlighting the security Quality of Service for WebSphere Application Server on the System z platform.

Table 5-1 Security highlights for WebSphere Application Server

WebSphere Application Server for Linux on System z	WebSphere Application Server for z/OS
► Full J2EE and Java2 security ► Web Services security ► Authentication with LDAP, custom or file registries ► External centralized authorization with JACC provider ► Single sign-on ► Security attribute propagation ► CSIv2 support for RMI-IIOP ► Encryption supported by hardware cryptography ► Hipersockets across LPARs	► Full J2EE and Java2 security ► Web Services security ► Authentication with LDAP, custom or file registries ► External centralized authorization with JACC provider ► Single sign-on ► Security attribute propagation ► CSIv2 support for RMI-IIOP ► Encryption supported by hardware cryptography ► Hipersockets across LPARs ► Authentication with centralized SAF-RACF ► External centralized authorization with SAF-RACF ► Advanced operating system provides isolation and protection ► Thread-level security using local connectors ► Security Access Facility (SAF-RACF) for user IDs, groups, roles, keyrings, certificates ► Built-in advanced auditing capabilities with SMF and SAF-RACF ► Accounting with SMF data

Table 5-2 lists features highlighting the availability Quality of Service for WebSphere Application Server on the System z platform.

Table 5-2 Availability highlights for WebSphere Application Server

WebSphere Application Server for Linux on System z	WebSphere Application Server for z/OS
► System z hardware redundancy ► Built-in High Availability manager ► Horizontal clustering of WebSphere Application Servers (ND topology) ► Vertical clustering of WebSphere Application Servers (ND topology)	► System z hardware redundancy ► Built-in High Availability manager ► Horizontal clustering of WebSphere Application Servers (ND topology) ► Vertical clustering of WebSphere Application Servers (ND topology) ► Redundancy of application execution JVM (multiple Servant regions) ► Sysplex Distributor and DVIPA for intelligent and dynamic load balancing ► Automatic Restart Manager (ARM)

Table 5-3 lists features highlighting the scalability Quality of Service for WebSphere Application Server on the System z platform.

Table 5-3 Scalability highlights for WebSphere Application Server

WebSphere Application Server for Linux on System z	WebSphere Application Server for z/OS
► System z hardware vertical scalability ► Horizontal clustering of WebSphere Application Servers (ND topology) ► Vertical clustering of WebSphere Application Servers (ND topology)	► System z hardware vertical scalability ► Horizontal clustering of WebSphere Application Servers (ND topology) ► Vertical clustering of WebSphere Application Servers (ND topology) ► Dynamic vertical scalability with WLM (automatic startup of Servant regions) ► Goal-driven workload management ► Workload Manager (WLM) and Intelligent Resource Director (IRD) for the coordination of shared resources ► Different performance goals depending on J2EE application with transaction classes ► Fast cross memory communications for local connectors to data (CICS, IMS, DB2)

Table 5-4 lists features highlighting the transactionality Quality of Service for WebSphere Application Server on the System z platform.

Table 5-4 Transactionality highlights for WebSphere Application Server

WebSphere Application Server for Linux on System z	WebSphere Application Server for z/OS
► Support for local and global transactions ► Coordination of XAResource-based resource managers	► Support for local and global transactions ► Coordination of XAResource-based resource managers ► Coordination of resource managers through RRS for transaction management and recovery ► Two-phase commit with CICS, DB2, IMS, and WebSphere MQ using local connectors

Table 5-5 lists features highlighting the manageability Quality of Service for WebSphere Application Server on the System z platform.

Table 5-5 Manageability highlights for WebSphere Application Server

WebSphere Application Server for Linux on System z	WebSphere Application Server for z/OS
► WebSphere Administrative Console, scripting interface (wsadmin), administrative framework (JMX™) ► Integrated Tivoli Performance Viewer and Advisor ► Network Deployment configuration for centralized management and clustering	► WebSphere Administrative Console, scripting interface (wsadmin), administrative framework (JMX) ► Integrated Tivoli Performance Viewer and Advisor ► Network Deployment configuration for centralized management and clustering ► WLM/RMF integration for classification of WebSphere workload (http, iiop, jms, sib) and reporting ► SMF Type 120 for WebSphere Java application capacity planning and charge back ► Well-established z/OS set of systems management tools and procedures (SMP/E, System Automation, RMF, TWS, and so forth)

With *AJAX (Asynchronous JavaScript™ and XML)*, the interaction model for Web applications has become more robust, like desktop applications, with continuous interaction and improved usability. Some of the benefits of adding AJAX capabilities to applications are a more interactive, differentiated experience, which can lead to longer sessions and increased customer loyalty and responsiveness; local actions, which can result in fewer abandoned transactions; higher completion rates; and higher end user productivity. Unlike other rich Web user interface approaches, AJAX applications leverage standard browser features and do not require browser plug-ins.

The WebSphere Application Server Feature Pack for Web 2.0 is an IBM-supported solution for creating AJAX-based applications and mashups on WebSphere Application Server. In addition to providing AJAX development tools, this feature pack includes server enhancements to support common Web 2.0 applications patterns. The Feature Pack for Web 2.0 includes Web 2.0 to SOA connectivity, AJAX messaging, and an AJAX development toolkit.

5.5.2 Choosing a new advanced Portal on System z

As a reminder, Presentation services can also provide advanced capabilities such as advanced personalization and customization, composite display of multiple applications, customization of the computing desktop to match individual work patterns and roles, and many other capabilities. These capabilities fall into the scope of what is called a *Portal*. Although these capabilities could be developed on top of a Web application server platform, it would require a lot of time and effort to develop, standardize, and maintain such a solution. The fastest and most cost-effective way to build the Presentation services is from an advanced Portal software package.

This solution is most appropriate when a large set of the Presentation services capabilities are required. For example, most Portal solutions include features from search and composite display of multiple applications to advanced user interface fly-out menus, page navigation, drag and drop. Portals allow the presentation space to be partitioned into smaller pieces called *portlets*. As applications are created or converted to service-based portlets, Portal administrators can aggregate this new and enriched content faster and more economically to get a more collaborative user interface.

Each portlet represents a business function that might otherwise have been buried inside a monolithic application. In the Portal context, an application is composed of a set of portlets — for example, the administrative functions related to customers (create, update, delete), the order entry functions, order pricing, a solvency check against an external server, and so forth. Every Portal user, depending on role, might be granted access to all or some or the portlets. And of course, portlets can be shared and used in more than one application. A portal is the front end of an SOA environment that gives flexibility at the display level.

The role of the Portal server as an aggregation tool for presentation is a key component of the SOA infrastructure. Initially, the Portal was mostly seen as a way to consolidate access to different Web applications according to user profiles and to provide a way to display information. Now with the advent of applications composed of portlets, the role of the Portal within the SOA infrastructure is emphasized. For these two roles, the need for high Quality of Service is strong: an enterprise cannot close the gateway to its information system.

The IBM products which satisfy these requirements are from the WebSphere Portal family. Running on System z, different flavors of this Portal server exist which differ in terms of capabilities and Qualities of Service:

- *WebSphere Portal Express, Server, Enable, Extend* which run on Linux on System z
- *WebSphere Portal Enable for z/OS*

WebSphere Portal relying on a WebSphere Application Server base can also benefit from the advanced infrastructure capabilities of WebSphere Extended Deployment which runs on Linux on System z and WebSphere Extended Deployment for z/OS for improved Quality of Service.

Portal solutions often have to access a huge amount of information. This can be information regarding user personalization, enterprise intranet content, or even work documents. With the large number of users accessing Portal solutions the quantity of information that must of served by the technical solution is significant. For this reason it is better to have the Portal aggregation engine, the datastore, and the disks very close to each other and with fast communication channels. WebSphere Portal Enable for z/OS along with DB2 z/OS data sharing are an ideal combination to provide fast I/O throughput.

Figure 5-10 on page 246 shows an example of a WebSphere Portal Enable for z/OS deployment in a cluster configuration. It shows how you can leverage Parallel Sysplex features such as DB2 data sharing.

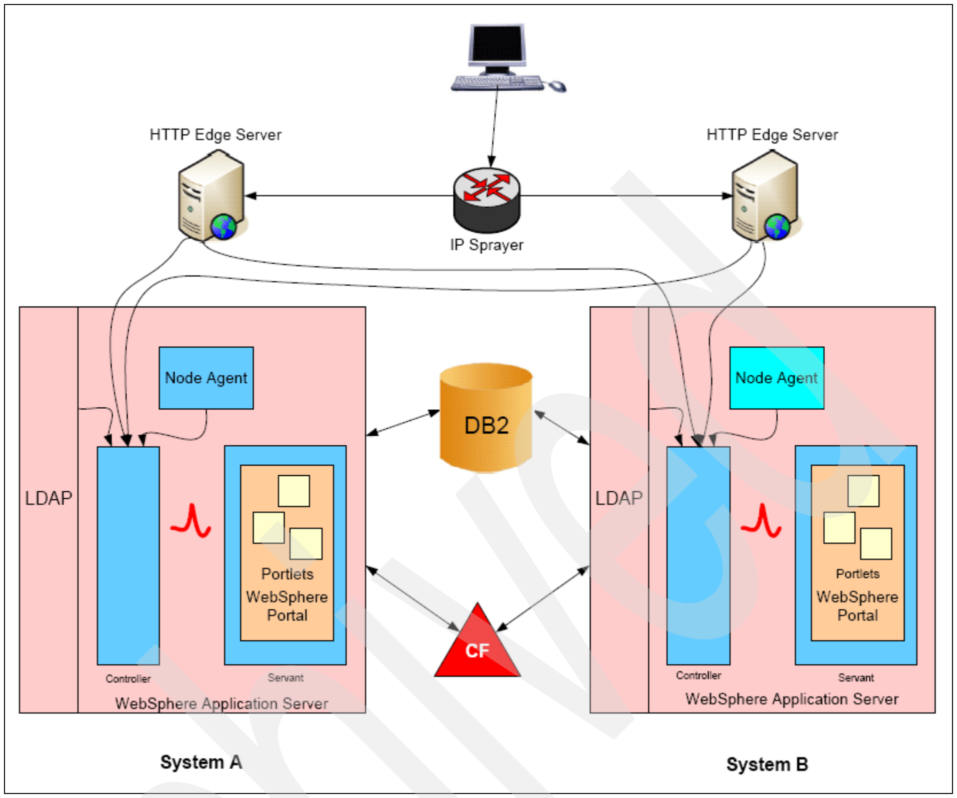


Figure 5-10 WebSphere Portal Enable for z/OS in a cluster configuration

Detailed functions of these products are well described on the IBM Web sites.

For integration in the larger SOA environment or more specifically with an ESB, WebSphere Portal relies on the rich WebSphere Application Server connector capabilities. It supports a large set of connectors and adapters that handle synchronous and asynchronous communication. Most of the connectors rely on connection pooling for high availability and high performance (Datasources, Connection Factories). WebSphere is fully compliant with the *Java Connector Architecture (JCA)*. WebSphere Application Server for z/OS is particularly well suited for integrating with Enterprise Information System with specific Quality of Service. For example, the use of local connectors to CICS, IMS, or DB2 allows for propagation of security identity and provides two-phase commit transactionality.

Because WebSphere Portal relies on the WebSphere Application Server base, it also leverages all the Quality of Service features of WebSphere Application Server. Section 5.5.1, “Choosing a new simple Web application server on

System z" on page 238 describes WebSphere Quality of Service highlights for both WebSphere Application Server on Linux on System z and on z/OS.

Section 5.4, "Platform choice criteria" on page 232 describes some requirements relevant for Interaction services that can help in determining the best platform for your purposes. We now go over some of these criteria and mention some solution components that can satisfy them.

Security criteria

- **Centralized management**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS can use SAF-RACF or LDAP and a JACC provider for centralized security management. SAF-RACF provides a highly available central security management solution for all subsystems (WebSphere along with DB2, CICS, IMS) running on a Parallel Sysplex. It can manage J2EE roles. In addition, Tivoli Access Manager can be a JACC provider or a central security management solution.
- **External authentication in a DMZ**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS or Linux on System z support external authentication in a DMZ using a *Trust Association Interceptor (TAI)*, or an *LTPA token* or a *custom login module*. Tivoli Access Manager WebSeal can be an RPSS that can be easily integrated with WebSphere. In addition, WebSphere DataPower can be used as an RPSS.
- **External authorization in the security manager**
  WebSphere Application Server for z/OS supports external authorization manager such as SAF-RACF or standardized JACC providers.
- **Single sign-on (SSO)**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS can manage Single Sign On Trust Association Interceptor, an LTPA token, or a custom login module. They can also do single sign-on with Collaboration services such as Lotus® Domino® using an LTPA token.
- **Identity propagation**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS provide identity propagation features. Depending on the communication transport, the identity propagation mechanisms vary. On z/OS a local connector to DB2, CICS, or IMS allows propagation of the RACF ID at the thread level using *Thread identity*. With RMI/IIOP *CSIv2* identity assertion can be used. With Web Services, *WS-Security* identity assertion, *UsernameToken,* or other *BinaryTokens* can be used. Also, not only does WebSphere support identity propagation, it also supports security attribute propagation in order to propagate not only an identity but also a complete

security context. *Tivoli Federated Identity Manager* can also help in the transformation of credentials from one security domain to another.

- **Support of standards**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS support WS-Security, J2EE security, JACC, SSL/TLS, and many others.
- **Minimal network communication**
  Because collocation is common on the z/OS platform, WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS make use of local optimized connectors with memory-to-memory communication to an ESB such as WebSphere Message Broker (*bindings* mode) or to Enterprise Information Systems such as CICS, IMS or DB2. Across z/OS LPARs and across Linux on System z partitions, *hipersockets* can be used for any TCP/IP traffic.
- **Highly secured truststore for keys**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS can use ICSF and System z cryptographic hardware to store keys (PKDS, CKDS). A master key stored in the hardware secures the keystore itself. Also RACF can be used as a keystore.
- **High isolation between environments**
  There is a high level of security between System z LPARs. PR/SM LPAR for the IBM System z9® 109 (currently z9 EC) was evaluated under the Common Criteria at evaluated assurance level 5 in Germany, level 4 worldwide. The certificate was published on March 24, 2006.
- **Auditing of all security events**
  RACF along with SMF can log all security audit events. This applies not only to the Interaction services deployed on z/OS but also to all the other services deployed on z/OS. This provides a central management solution for security audit events.

Availability

- **Clustering of software instances**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS support horizontal and vertical clustering for high availability. Unique on z/OS is the ability to start and run multiple servant regions, producing even better availability. With WebSphere clustering, workload balancing is set up easily using the WebSphere plug-in and synchronization is handled conveniently with the *Data Replication Service (DRS)*.
- **High availability of the hardware**
  Redundancy is perhaps the most prevalent mechanism to ensure resiliency, and is a key theme throughout the System z platform. System z designers have worked through the years to eliminate any single point of failure. The platform has many redundant components. For example, there are two power

feeds that enable the system to survive the loss of one. There are two MRUs (Modular Refrigeration Units). The service subsystem, internal battery, oscillator, and all of the processors are redundant as well.

- **Workload balancing mechanisms based on performance goals**
  z/OS WLM can help distribute workload across multiple system images in a Parallel Sysplex. Sysplex Distributor checks with WLM for a recommendation on where to route a new work request. WLM checks the workload on each system image and returns a list of eligible servers that can take care of the new work. The highest server in the list is the preferred server. The objective is to send work where there is available capacity or, if there is no available capacity, where the least important work is running. Dynamic workload distribution can also be provided by WebSphere Extended Deployment features.
- **Robust data synchronization mechanisms**
  WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS support synchronization of stateful application data using the Data Replication Service or using persistent sessions in DB2 z/OS.
- **Sharing of the datastore**: WebSphere Portal Enable for z/OS can use DB2 on z/OS to store all its data, and leverages DB2 data sharing capabilities for robust data synchronization within the multiple DB2 members. DB2 on z/OS data sharing allow for workload sharing across multiple DB2 members even when accessing the same data.
- **Seamless hardware and software upgrades**
  Today, the System z platform eliminates planned outages with its capability to concurrently change hardware without affecting the application. The concurrent hardware maintenance capability helps System z to provide service to its users at any time, 24x7, without scheduled outages for maintenance of system and data.
- **Automatic restart of a failed instance**
  z/OS *Automatic Restart Manager (ARM)* is capable of restarting any z/OS subsystem, such as a WebSphere instance, when it detects that it is not running anymore.

Performance

- **Workload management based on performance goals:** The unique workload and self-management capabilities provided by z/OS *Workload Manager (WLM)* and the *Intelligent Resource Director (IRD)* enable z/OS to handle unpredictable workloads and meet response goals through effective use of CPU and I/O resources with minimal human intervention for setup and operation, making it the most advanced self-managing system. WebSphere Portal Enable for z/OS, running on top of WebSphere Application Server for z/OS, make full use of WLM and IRD services.

- **Assigning specific performance goals for transactions:** WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS along with z/OS WLM provide the capability to assign different performance goals to different transactions deployed to the same server. This allows you to prioritize transactions depending on their goals.
- **Proximity to data and business services:** On z/OS, Presentation services and data access and business application services can be co-located on the same hardware. Optimized local connectors or JDBC drivers can be used that are using memory-to-memory communication.
- **Encryption:** System z possess a wide range of cryptographic hardware for higher performance from CP Assist for Cryptographic Function (CPACF) to Crypto Express2 coprocessor and accelerators.
- **Dynamic scalability:** WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS provide the capability to start additional execution JVMs (servant regions) based on WLM performance goals. The number of servants can be managed dynamically by WLM, which can increase or decrease the number of servants in response to changing workloads. In the event of an application failure that takes a servant down, WLM will initiate a restart of another servant. Even if you have only one, non-clustered Portal server, with WebSphere Portal on z/OS, you can have multiple servants, which gives you protection against the most common application problems that cause servant JVM failure. Dynamic scalability can also be provided by WebSphere Extended Deployment.

Transactionality

- **Two-phase commit with distributed services:** WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS support coordination of XAResource-based resource managers.
- **Two-phase commit with Enterprise Information Systems services**: WebSphere Application Server for z/OS and WebSphere Portal Enable for z/OS support coordination of resource managers through *Resource Recovery Services (RRS)* for transaction management and recovery with CICS, IMS, DB2, and WebSphere MQ.
- **Accounting resource usage tracking:** SMF/RMF collect information about the behavior of the system and the work that runs on that system. More precisely, SMF Type 120 is used for WebSphere applications capacity planning and charge back.

5.5.3 Extending an existing presentation solution to System z

If the decision has been made to run some Presentation services on the distributed platform or if Presentation services run on the distributed platform because there were no high QoS requirements so far, then it makes sense to analyze the platform choice in the following cases:

- Some new Presentation services are being deployed and both functional and non-functional requirements have to be listed and analyzed for the platform choice. The considerations mentioned in “Platform choice criteria” on page 232 can help in the design process.
- Some existing Presentation services have new users, new usage, or new requirements that fall in the criteria listed in “Platform choice criteria” on page 232.

Depending on the functional and non-functional requirements for the new Presentation services, or new users or new usage, extending the presentation solution to System z can satisfy the new needs. Either the new presentation functions can be deployed on System z, or the same presentation functions but for a different group of users or different applications can be deployed on System z. shows The patterns available when federating or extending Interaction services between platforms are described in 5.3.4, “Interaction services federated on distributed and z/OS” on page 229.

Extending the Presentation solution to System z can be achieved in several ways:

- A complete Portal solution with all portlets can be deployed on both the distributed and System z platform and users are directed to one or the other depending on some criteria.
- A finer granularity can be obtained by deploying specific portlets to the distributed platform and some others to the System z platform.

There can be a strategy to keep the distributed Presentation services as the front end for all requests with remote calls to Presentation services to System z Presentation services. This has the disadvantage of being dependent on the Quality of Service of the distributed front-end, even for the Presentation services deployed on System z. Alternatively, requests can go directly from the end-user to both distributed services or System z services depending on the chosen services.

Also from a service perspective, different integration approaches can be chosen for federation. Portlets can be published as Web Services or Presentation integration techniques can be used.

Using WSRP services

WebSphere Portal Enable for z/OS supports the *Web Services for Remote Portlets (WSRP)* standard. By using this standard, Portals can provide portlets, applications, and content as WSRP services, and other Portals can integrate the

WSRP services as remote portlets for their users. The WSRP standard and specification is provided by OASIS. It defines a Web service communication interface for interactive presentation-oriented Web services. This standard simplifies the integration of remote portlets, applications, and content into Portals. With WSRP, Portal administrators can select from a rich choice of remote content, portlets, and applications, and integrate them into their Portal with just a few mouse clicks and no programming effort. Providers and consumers use this interface for providing and consuming the Web services. The WSRP services appear and operate to Portal users exactly like local portlets.

Portlets can now be run remotely from a Portal. This allows the Portal infrastructure to become more distributed. It can include several Portals that run on various platforms, including the System z platform for higher Quality of Service.

Figure 5-11 illustrates this architecture.

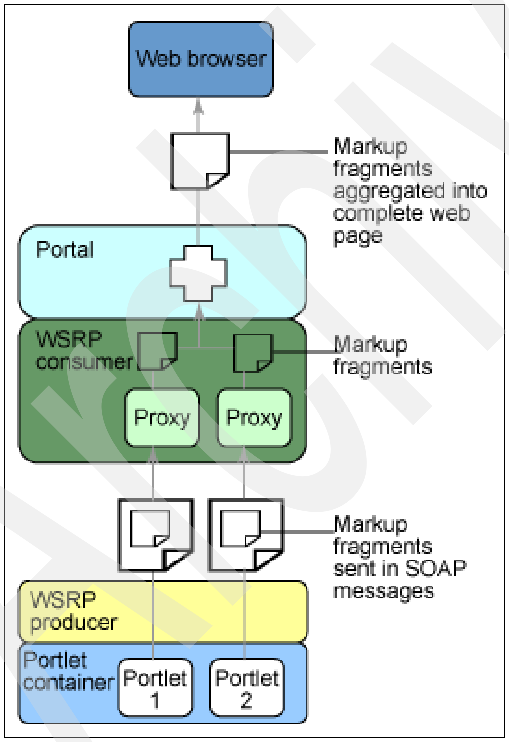


Figure 5-11 Portal integration using WSRP

Presentation integration

Presentation integration represents the simplest method of incorporating content into a WebSphere Portal deployment and is based solely upon the ability to screen scrape, either through the deployment of an iFrame or *Web Clipping* portlet, or existing visual content served by one or more back-end servers. This approach, however, has the severe drawback that content cannot be personalized or manipulated in any shape or form. Furthermore, in terms of overall performance, presentation integration does not normally sit well with enterprise scale deployments due to the lack of any type of brokerage or connection pooling mechanism for reducing the amount of back-end requests. For example, a Portal page containing two iFrame portlets will result in two separate back-end calls for a single Portal page request. This is irrespective of whether the content is served by the same back-end server or not.

The following options are available for Presentation integration:

- Using the *Web Page portlet*: Web Page allows you to include data from an external file or Web site directly on a Portal page. The Web page displayed by the portlet is retrieved directly from the URL (that is, it is not cached), making this portlet useful for viewing real-time data. Conversely, however, because the data displayed by the portlet is a complete Web page, formatted for an entire browser window, sometimes the layout of the page can be compromised by the smaller size of the portlet window in which it is displayed. It can handle authentication. This portlet uses iFrames, an enhancement that is incompatible with older Netscape browsers (versions 4.x and earlier). It works on Microsoft Internet Explorer® 5.0 and later, as well as newer versions of Mozilla (1.5 and greater).

  Support of the IBM Web Page portlet might be removed in a future release of WebSphere Portal. Use of the IBM Web Clipping portlet is recommended instead. Currently, the IBM Web Clipping portlet supports features available with the Web Page portlet, including support to include and retrieve data for display in a portlet from an external URL defined file or Web site.

- Using the *Web clipping portlet*: Many Web sites and browser-based applications are designed for desktop displays that support large windows and complicated layout. Depending on the amount and complexity of the content, it can be impractical to include the content in a portlet. If the users also expect to access the portlet from a mobile device, such as a phone or personal digital assistant (PDA), the ability to tailor content to these small displays is increasingly important.

  Web clipping portlets can identify and extract specific portions of a document for display in a portlet. In this way, it keeps those pieces of information that are important, while discarding undesirable information or elements in the document that the client device is incapable of displaying. The Web clipping process simplifies what is being sent to the device, while reducing the amount of data being transmitted, which is a particularly useful aspect when wireless devices are involved.

- Adding a remote *RSS feed*: The RSS portlet allows you to include formatted data from external *Really Simple Syndication (RSS)* feeds. An RSS feed is simply an XML file in a standardized format, often used for news stories or any other syndicated information. By pointing the RSS portlet at a particular feed, the portlet will display the feed's most up-to-date content every time the portal page is refreshed. For example, if you select a BBC news feed (see related information), whenever that RSS XML file is updated on the remote site, the portlet will reflect those updates on the next portal page reload.

5.5.4 Integrating with other services

In this section, we discuss technology choices for connectivity from the Interaction Presentation services.

When considering the various types of integration applicable to a WebSphere Portal deployment, it is also often helpful to understand which type of connectivity best suits the actual approach. It is also worth remembering that non-technical factors, such as available skill sets, and standards within an organization, can influence the choice of a particular type of connectivity.

- **Web Services:** Web Services are based on an open-standards way of defining and invoking a service. The implementation of the requestor and provider are hidden from each other, allowing portability in implementation. The coupling is based on the service interface, and a variety of transport protocols can be used. Both synchronous and asynchronous communication is possible, but each service defines the mode it supports. The basic stack is comprised of HTTP, XML, SOAP, WSDL, UDDI, and WSFL. Web Services can employ XML as an encoding schema that is widely adopted. They are relatively “heavy” to implement, and are best suited to inter-enterprise communication, or adopted as an enterprise-wide standard for leveraging an ESB, for example. Web services are not built to be high performing, so are not suitable for transactions that require very large throughput.
- **Messaging**: Messaging interfaces such as WebSphere MQ and JMS are based on the asynchronous exchange of messages between producers and consumers. *Point-to-Point* and *Publish-Subscribe* communication patterns are provided. Messages are placed on a queue by the sending application, and those messages are then consumed by a receiving application. With messaging, you take advantage of a simple and common API. You adopt industry-standard programming models and you make these available on a selection of operating systems. Messaging provides assured delivery for business-critical information. Messaging provides asynchronous (as well as

synchronous) processing for loose coupling of applications and control of the rate at which information is processed.

- **Adapters:** *Adapters* provide access to business logic in a tightly coupled manner. An adapter is specific to a particular Enterprise Information System (EIS) and generally requires client code to be written to parse the proprietary format of the data provided by the EIS. However, this tight coupling allows an adapter to map security, transaction information, and other Quality of Service information between the client and the EIS based on the well-established capabilities of EIS gateways. While adapters typically provide a synchronous interface, the latest specifications define an asynchronous mode as well, and some adapters implement this mode.

The following recommendations are made with regard to the selection of the most appropriate connectivity technology:

- Use Web Services when portability or interface standardization is a prime concern.
- Use Messaging when high QoS and loose coupling or asynchronous invocation is needed.
- Use JCA adapters when high QoS and synchronous invocation are needed.

5.5.5 Integrating with Process services

Portals are the ideal interface for the human component of business processes. For example, Portals can display visual prompts to notify users when they are assigned a new task. These prompts link users to their list of tasks, where they can view, claim, and launch tasks on a new Portal page. Each task is part of the larger business process, which can include any combination of human activities with automated services, creating a complete end-to-end workflow solution.

Because of the WebSphere Application Server common base, WebSphere Portal and WebSphere Process Server have several connectivity options from RMI/IIOP, to JMS or SOAP, calling specific WebSphere Process Server APIs or facades. Figure 5-12 on page 256 shows some possible connectivity options.

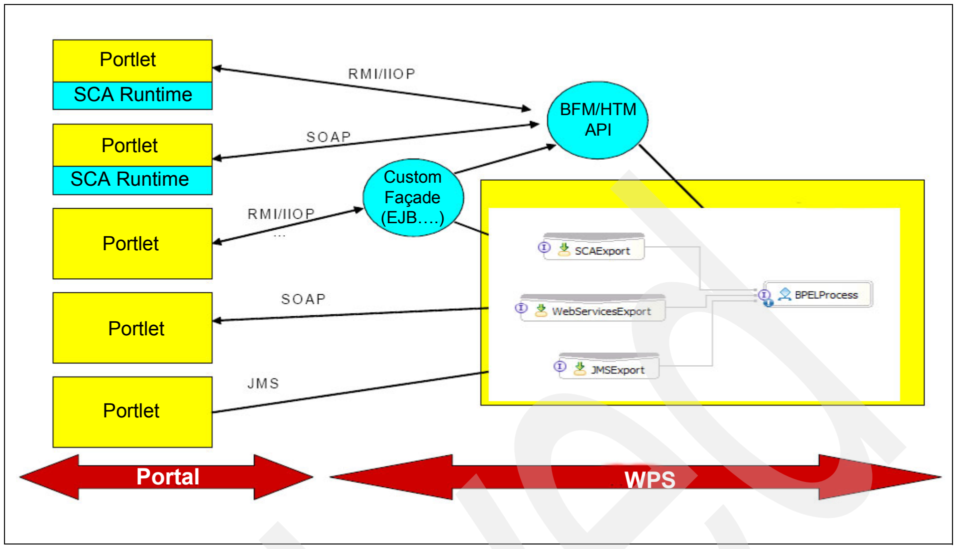


Figure 5-12 WebSphere Portal integration options with WebSphere Process Server

5.5.6 Integrating with back-end systems on System z

Integrating with CICS

WebSphere Application Server, and this is even more true on z/OS, has developed a lot of capabilities for integrating with Enterprise Information Systems such as CICS.

WebSphere Portal leverages these capabilities and provides various synchronous and asynchronous mechanisms to communicate with CICS. This can be done with Web Services (SOAP), messaging middleware such as WebSphere MQ, or using an adapter such as the CICS Transaction Gateway (CICS TG).

Figure 5-13 on page 257 shows several connectivity options between Presentation solutions like WebSphere Application Server or WebSphere Portal and CICS.

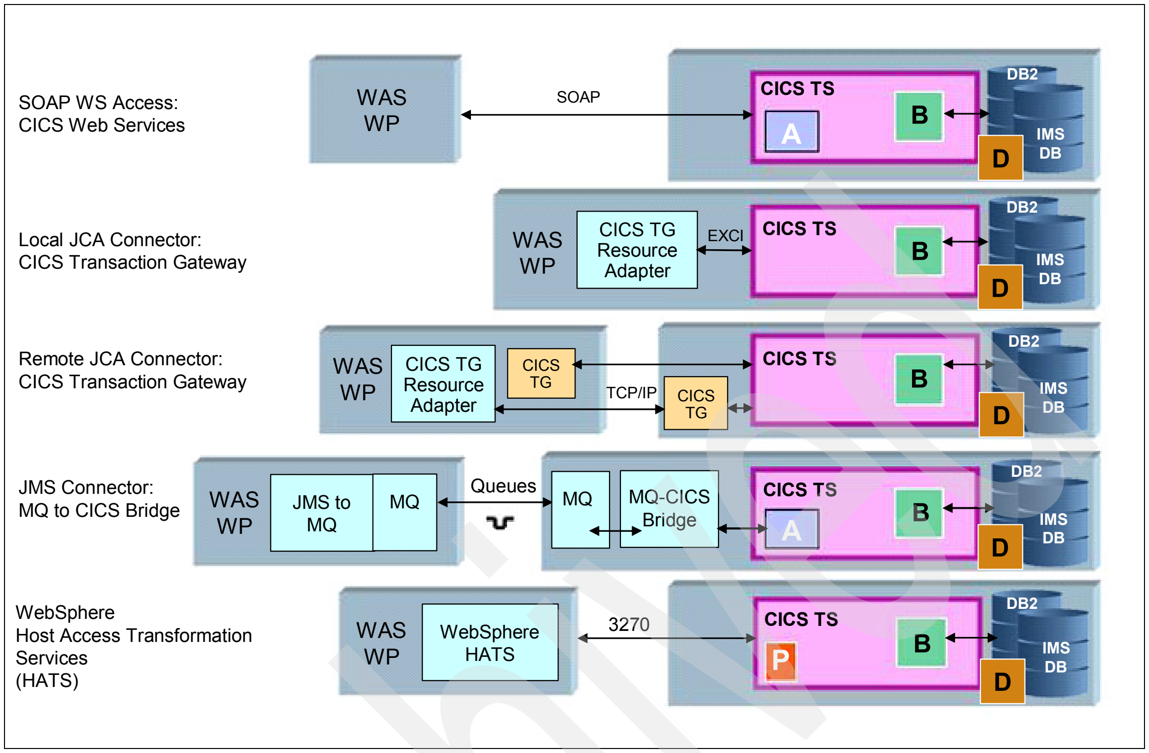


Figure 5-13 Connectivity solutions from WebSphere Application Server or WP to CICS transactions

More discussion about these solutions can be found in the IBM Redbooks publication *WebSphere for z/OS V6 Connectivity Handbook,* SG24-7064.

Integrating with IMS

WebSphere Application Server, and this is even more true on z/OS, has developed a lot of capabilities for integrating with Enterprise Information Systems such as IMS.

WebSphere Portal leverages these capabilities and provides many synchronous and asynchronous mechanisms to communicate with IMS transactions. This can be done with Web Services (SOAP), messaging middleware such as WebSphere MQ, or using an adapter such as IMS Connect.

Figure 5-14 on page 258 shows several connectivity options between Presentations solutions like WebSphere Application Server or WebSphere Portal and IMS transactions.

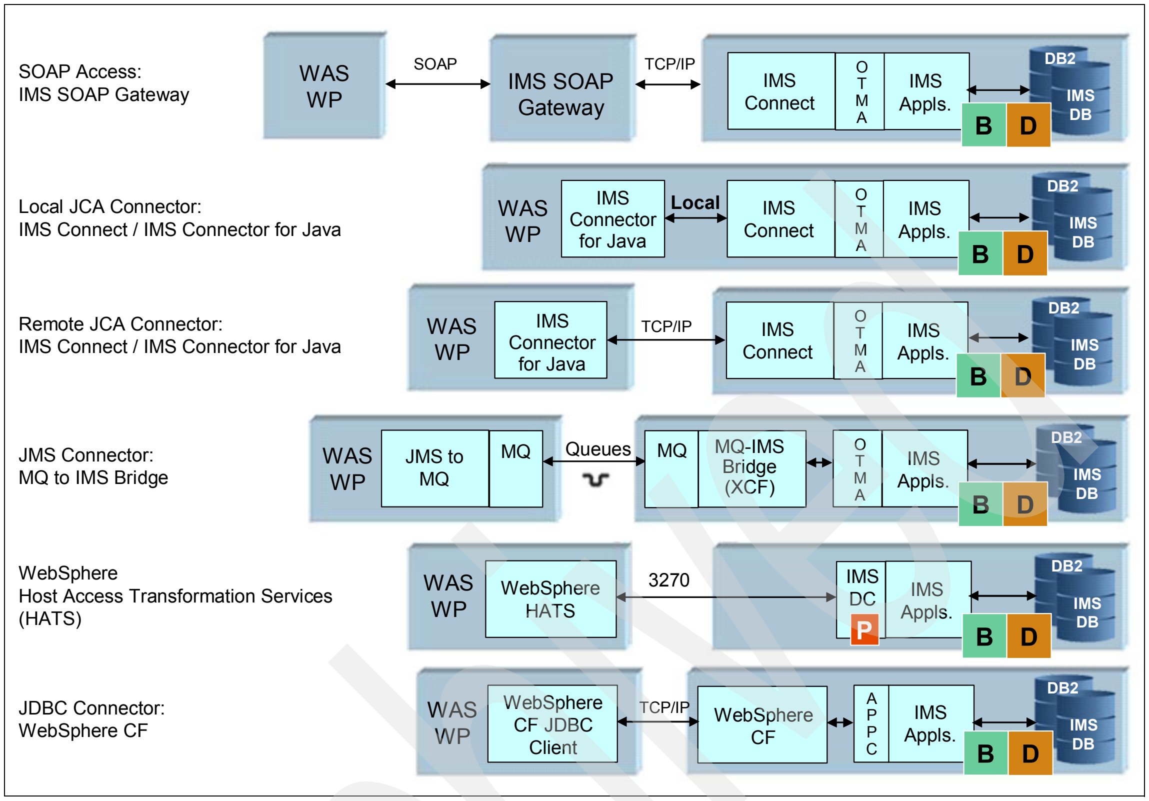


Figure 5-14 Connectivity solutions from WebSphere Application Server or WP to IMS transactions

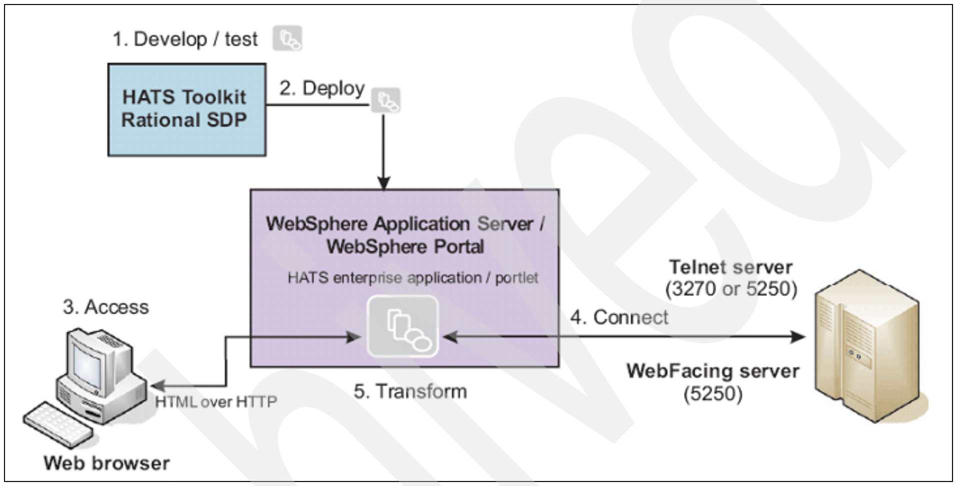
More discussion about these solutions can be found in the IBM Redbooks publication *WebSphere for z/OS V6 Connectivity Handbook,* SG24-7064.

Refacing CICS 3270 presentation

“Refacing” of an application refers to the process of applying some form of transformation via middleware to provide a new user interface to an existing application. This transformation might, for example, transform a 3270 datastream into HTML. This transformation allows the refacing of an application, but the underlying application remains unchanged. This is beneficial if both a true 3270 channel as well as the refaced channel must be maintained in parallel. Thanks to refacing middleware, a single application can service two or more user interface channels without modification. At first glance this might seem like a satisfactory solution. However, the fact that the existing application remains unchanged provides less flexibility in the presentation services because both the 3270 services and the transformation services need to evolve at the same time.

Rational Host Access Transformation Services (HATS) provides both development and runtime components in order to assist in the transformation of 3270 type applications to have a variety of interfaces. The HATS development tooling is provided as an Eclipse plug-in to Rational Application Developer (RAD). RAD allows you to reface the application as a Web interface or as a Web Service interface. Also recently announced is the ability to build portlets to be deployed in WebSphere Portal server.

HATS process flow is illustrated in Figure 5-15.



1. Develop / test
HATS Toolkit Rational SDP
2. Deploy
WebSphere Application Server / WebSphere Portal
HATS enterprise application / portlet
3. Access
4. Connect
Telnet server (3270 or 5250)
WebFacing server (5250)
HTML over HTTP
5. Transform
Web browser

Figure 5-15 IBM Rational Host Access Transformation Services

HATS Web applications, including portlets, can be developed with an interface that matches a company's Web or portal pages, and users can access them through Web browsers, including supported Web browsers on mobile devices. HATS rich client applications can be developed to run in an Eclipse *Rich Client Platform (RCP)* implementation or in the *Lotus Expeditor Client* to provide native client applications targeted for a user's desktop.

The starting point in application refacing within RAD is to import BMS or MFS maps. From these 3270 maps it is possible to select input and output fields as well as the mapping of any user interactions.

As just explained, CICS 3270 presentation solutions can be refaced using HATS. Also *CICS Web Support* can be used for the same purpose.

CICS Web support is a set of resources supplied with CICS TS for z/OS that provide CICS with a subset of the HTTP serving functions found in a general-purpose Web server. This allows CICS applications to be invoked by and reply to HTTP requests. A summary of how a CICS application can be Web-enabled using CICS Web support is illustrated in Figure 5-16.

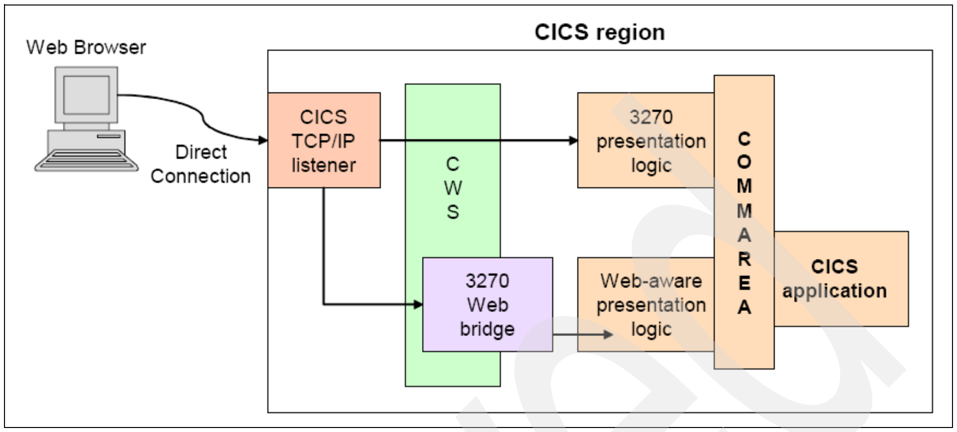


Figure 5-16 CICS Web Support components

CICS Web support provides a native HTTP interface to CICS; this interface can be used by both 3270-based transactions and applications that provide a callable COMMAREA interface. Two different configurations can be used to route the HTTP requests into the CICS region. Both configurations allow the use of the same facilities in CICS, although the configuration of the two options is significantly different.

CICS Web support can be used to invoke 3270 transactions. Then the facilities of the 3270 bridge are used. The 3270 transaction remains unchanged, and the 3270 output is converted to HTML. This function is known as the 3270 Web bridge.

Refacing IMS 3270 presentation

As explained before, IMS 3270 presentation solutions can be refaced using Rational Host Access Transformation Services (HATS). HATS is described in the preceding section.

IMS MFS Web enablement also can be used for the same purpose.

This feature allows an existing MFS-based IMS application to be refaced with an HTML-based presentation layer. The development time tooling comes in the form of the standalone MFS Web enablement utility. This utility allows for the importing of MFS source that in turn produces Java classes (as part of .war files) and XML Metadata Interchange (XMI) files. The resulting packaged .war file should then be deployed to a WebSphere Application Server. At run time the WebSphere Application Server application using the MFS instance servlet, MFS servlet, and MFS adapter will allow for the conversion of an HTML Web page to a 3270 bytestream understood by the IMS MFS-based application. In the response the bytestream is then used in the generation of a dynamic Web page.

Figure 5-17 shows MFS Web enablement components and flows including WebSphere Application Server, IMS Connector for Java, and IMS Connect.

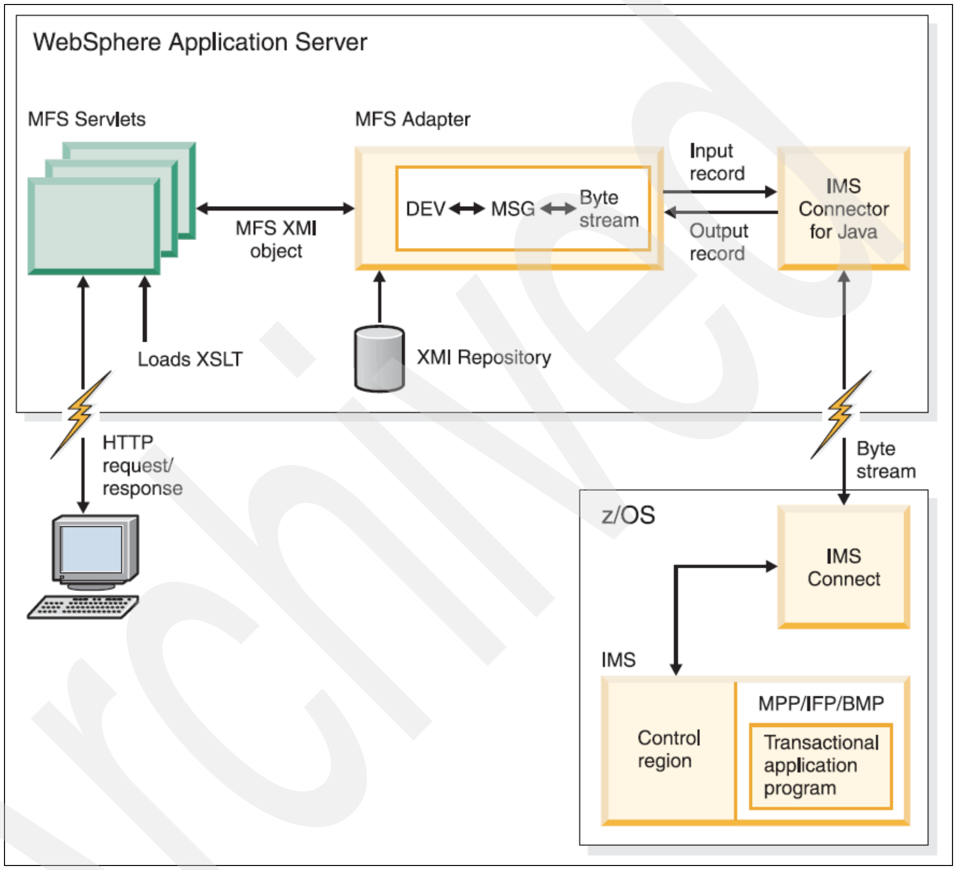


Figure 5-17 MFS Web enablement components and flows

5.6 Collaboration solutions

As a reminder, Collaboration services can provide capabilities such as e-mail, teamroom applications, team spaces, Instant Messaging, Wikis, Web conferencing, and so forth. There are many software and hardware solutions in the marketplace with different scopes to satisfy these functional needs.

In this section, we describe the Collaboration solutions pertinent to the System z platform.

5.6.1 Choosing a new Collaboration solution

One of the most prominent Collaboration service in corporations is the e-mail messaging server. It has become more and more important for conducting business and communicating among employees of the same company and also across corporations. Cutting-edge e-mail solutions have a rich set of features such as calendar and scheduling, discussion databases, team spaces, close integration with instant messaging, and so forth.

The IBM products that satisfy these requirements are from the Lotus Domino family.

Lotus Domino software provides world-class collaboration capabilities that can be deployed as a core e-mail and enterprise scheduling infrastructure, as a business application platform, or both. Lotus Domino software and its client software options deliver a reliable, security-rich messaging and collaboration environment that helps companies enhance the productivity of people, streamline business processes, and improve overall business responsiveness.

Among the many benefits provided by Lotus Domino, it:

- Extends messaging with built-in collaboration tools
- Offers flexibility and choice in hardware platform, operating system, directory, and client access
- Provides industry leading security features to help safeguard business-critical information
- Can help to reduce the Total Cost of Ownership (TCO) by efficiently using CPU resources, network bandwidth, and disk storage
- Maximizes server availability with advanced clustering, transaction logging, server fault recovery, and automated diagnostic tools
- Helps reduce time and costs associated with deploying and managing the infrastructure, through advanced administration features
- Supports Web services and open standards and offers tools for integration with existing applications

There are two ways to run Domino on a System z platform: either on the z/OS operating system, or on the Linux on System z operating system (with or without z/VM). It is helpful to consider how the characteristics of these two operating systems differ as far as Domino is concerned. We have listed these under Quality of Service and cost:

- **Quality of service (QoS)**: By Quality of Service, we mean the standard of service that you are able to deliver to the users, including reliability, availability, serviceability (RAS), scalability, security, and manageability. Linux and z/OS differ in these areas significantly:
  - While both operating systems benefit from the Quality of Service features of the System z hardware, Domino on z/OS exploits more of the hardware features, such as use of the hardware cryptographic facility that improves SSL performance.
  - z/OS contains many advanced functions for running mission-critical applications. Half of the operating system kernel is there for error recovery, so the z/OS operating system seldom fails. Many administration tasks can be done without taking the operating system down.
  - z/OS contains many advanced features for automated system management, including advanced workload management with Intelligent Resource Director (IRD), Resource Measurement Facility (RMF) for performance monitoring, and System Management Facility (SMF) for resource consumption measurement.
  - Domino on z/OS is proven to run thousands of Domino users in a single LPAR under z/OS.
- **Cost**: A second consideration is cost. This book cannot provide detailed cost comparisons, but we suggest you keep in mind the following points:
  - Hardware costs depend on detailed sizing information that is not available at the time of writing. IBM offers IFLs for Linux, in addition to standard System z processors. Disk costs are likely to be the same in both options. Keep in mind that hardware costs are likely to be less than one-third of the total cost of running the solution over five years.
  - Software costs might not differ between z/OS and Linux as much as you would expect. Domino is chargeable in both environments. While the z/OS cost is likely to be higher than Linux, options such as *System z New Application License Charges (zNALC)* can reduce that cost on LPARs where you are running a qualified "new workload" application like Domino. You will pay for a Linux distribution, and will also probably pay for Linux support. In both cases you will need some additional software, such as for backup.
  - z/OS costs can be limited by using zNALC or *Workload License Charge (WLC)*, or by running Domino on a separate System z server from other

System z workloads to avoid having to license unnecessary products on Domino capacity. On Linux, other software costs can be minimized by using IFLs, because if you use standard processors, you might again incur additional costs for other products on z/OS on the same machine.

– People and support costs are likely to be the highest cost that you incur in running the Domino service. The additional hardware and software costs in z/OS provide you with a great deal of extra function that will reduce your support effort and therefore your personnel costs.

– Also consider the costs of training and building up experience. Using skills that you have today will avoid the costs of developing or bringing in new skills. If you do not have skills today, Linux skills are more readily available in the market than z/OS skills.

In short, z/OS offers the ultimate in System z reliability and scalability. You can choose to run Domino on either z/OS or on Linux by considering a number of factors, especially what skills you have available.

Figure 5-18 shows a Domino infrastructure running on multiple Linux on System z guests.

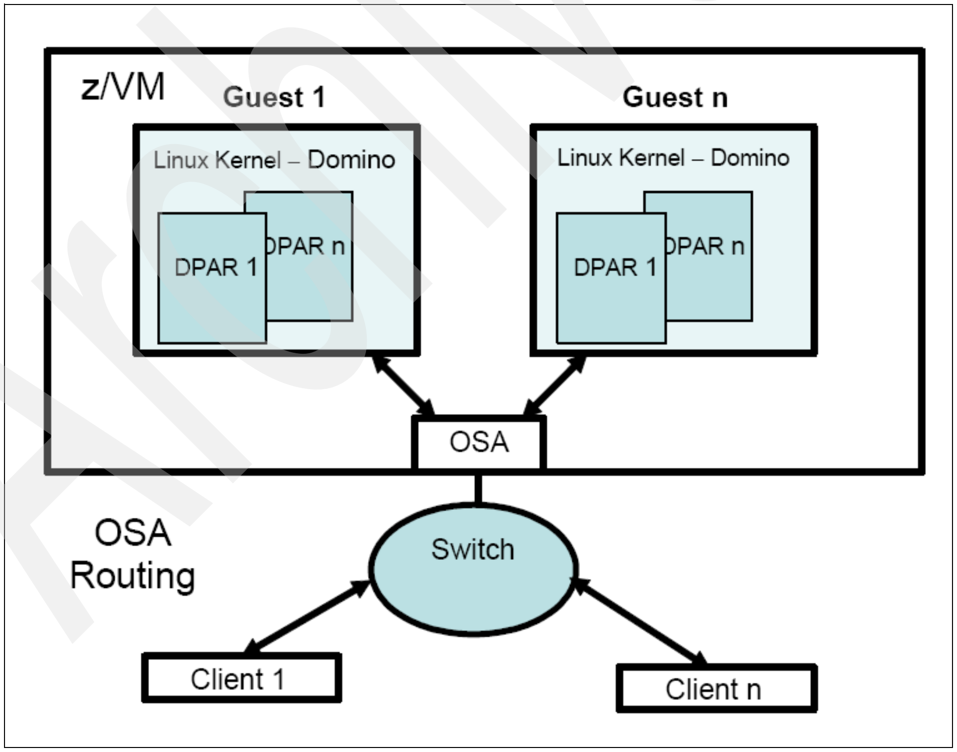


Figure 5-18 Multiple Domino partitions on multiple Linux on System z guests

5.6.2 Extend or federate the Collaboration solution with System z

If the decision has been made to run some Collaboration services on the distributed platform or if Collaboration services run on the distributed platform because there was no high Quality of Service requirement so far, then it makes sense to analyze the platform choice in the following cases:

- Some new Collaboration service are being deployed and both functional and non-functional requirements have to be listed and analyzed for the platform choice. The considerations mentioned in “Platform choice criteria” on page 232 can help in the design process.
- Some existing Collaboration service have new users, new usage, or new requirements that fall in the criteria listed in “Platform choice criteria” on page 232.

Depending on the functional and non-functional requirements for the new Collaboration services, new users, or new usage, extending the Collaboration solution to System z can satisfy the new needs. Either the new Collaboration functions can be deployed on System z, or the same Collaboration functions but for different group of users or different applications can be deployed on System z. Section 5.3.4, “Interaction services federated on distributed and z/OS” on page 229 shows different patterns when federating or extending Interaction services between platforms.

Extending the Collaboration solution to System z can be achieved in several ways:

- A complete Collaboration solution with all features can be deployed on both distributed and System z platform and users are directed to one or the other depending on some criteria.
- A finer granularity can be obtained by deploying specific features to the distributed platform and others to the System z platform.

In this discussion we use the Lotus Domino products as an example of a Collaboration solution. Lotus Domino is full-functioned and platform-independent across all platforms. The Domino source is made up of multiple layers of code. The top-most layers of code are platform-independent and are developed on whichever architecture the developer is most familiar with and tested on all architectures. These layers use no platform-specific code. There is the OS-specific layer. This layer consists of a platform-independent API that is coded and created for each supported architecture of Domino, and gives Domino access to platform-dependent features, but with platform-neutral behavior.

Because Lotus Domino is full-functioned and platform-independent, new Domino instances can easily be federated into an existing Domino domain. Therefore, new Domino for z/OS servers or Domino on Linux on System z servers can be seamlessly added to an existing Domino domain. The Domino infrastructure then allows you to administer resources and where they are deployed. For example, critical mailboxes or teamrooms requiring specific QoS can be moved or replicated from distributed Domino servers to Domino for z/OS servers.

One of the most prominent additions to application development starting in Domino 7 is Web services support. Domino 7 and later supports Web Services as a Web service provider and allows customers to increase the value of their Domino applications and data, allowing Domino to participate in service-oriented architecture.

6

Information services

In this chapter, we describe Information services and their role in a service-oriented architecture in a System z context.

This chapter discusses the following topics with regard to Information services:

6.1 Definitions, scope, and functions

6.1.1 Definitions

Information services contain the data logic of your business design. This logic exists on two levels. On the surface, information services provide access to the persistent data of your business. This can include query statements for retrieving the information you care about or referential integrity checks on the information manipulated by these services. These data services are available to the business application as services, often constructed with specific domain model semantics so that they appear for all intents and purposes as business application services.

Information services at this level incorporate the idea of federating multiple data sources, projecting logical views over those multiple sources to render a simpler access pattern for the service composition that needs it.

On another level, information services has its own sub-architecture for managing the flow of data across the organization. This is intended to handle two important requirements within the information management space: *data composition* and *data flow*.

The first of these is the need to compose information in a way that matches the composition of services in the business design. This is somewhat analogous to the kind of re-factoring that can occur with legacy applications to get them to fit better with the business design. However, it can also go deeper than that. It is fairly common practice to separate the database design from the application design and this is typically needed to achieve the level of performance and scalability required in many enterprise computing environments.

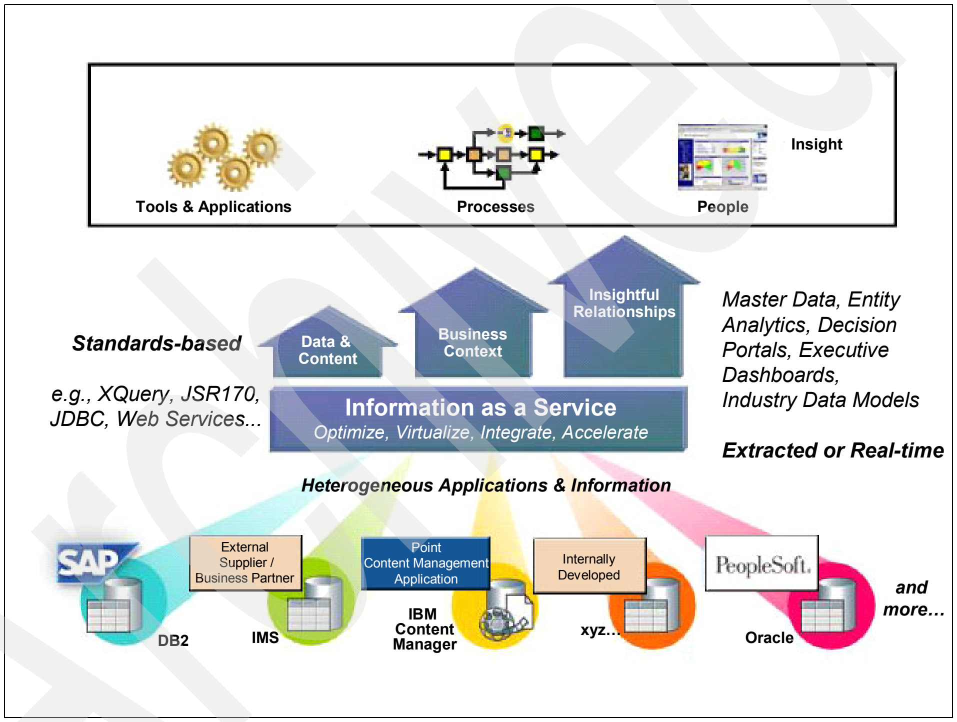
This also means that an application might have to code complex joins across the database design (that is, queries across multiple database tables, possibly defined by multiple schemas) in order to get the information it needs to perform its function. The complexity of these access patterns is exacerbated when collecting data needed for a complex composition of business services.

Even more than this, in a sufficiently large and heterogeneous system, where a business process might be composing services from many different parts of the information system, there is a good chance that the data for that composition will be coming from many different databases and types of data systems, some of them relational databases, different database catalogs, file systems, XML repositories, and so forth.

The second requirement addressed by the information services sub-architecture is the movement of information from one part of the enterprise to another as needed to satisfy its own data flow and life cycle requirements.

This might involve the use of *Extract-Transform-Load (ETL)* mechanisms to process and enrich data in bulk, batch processing activities involved in bulk transaction processing, and migrating data from master-data-of-record databases to information warehouses that can be used to perform postprocessing and business intelligence and content management functions, which in turn are made available to the business application as services.

Figure 6-1 illustrates the concept of *Information as a Service*.



Tools & Applications
Processes
People
Insight
Standards-based
Data & Content
Business Context
Insightful Relationships
Master Data, Entity Analytics, Decision Portals, Executive Dashboards, Industry Data Models
e.g., XQuery, JSR170, JDBC, Web Services...
Information as a Service
Optimize, Virtualize, Integrate, Accelerate
Extracted or Real-time
Heterogeneous Applications & Information
SAP
External Supplier / Business Partner
Point Content Management Application
Internally Developed
PeopleSoft.
DB2
IMS
IBM Content Manager
xyz...
Oracle
and more...

Figure 6-1 Information as a Service

6.1.2 Scope

The purpose of Information services is to manage diverse data and content in a unified manner. Information services provide the capabilities necessary to federate, replicate, and transform disparate data sources. Publishing consistent, reusable services for information processing makes it easier for processes to get the information they need from across a heterogeneous landscape.

Figure 6-2 shows the place of Information services in the IBM SOA Reference Architecture.

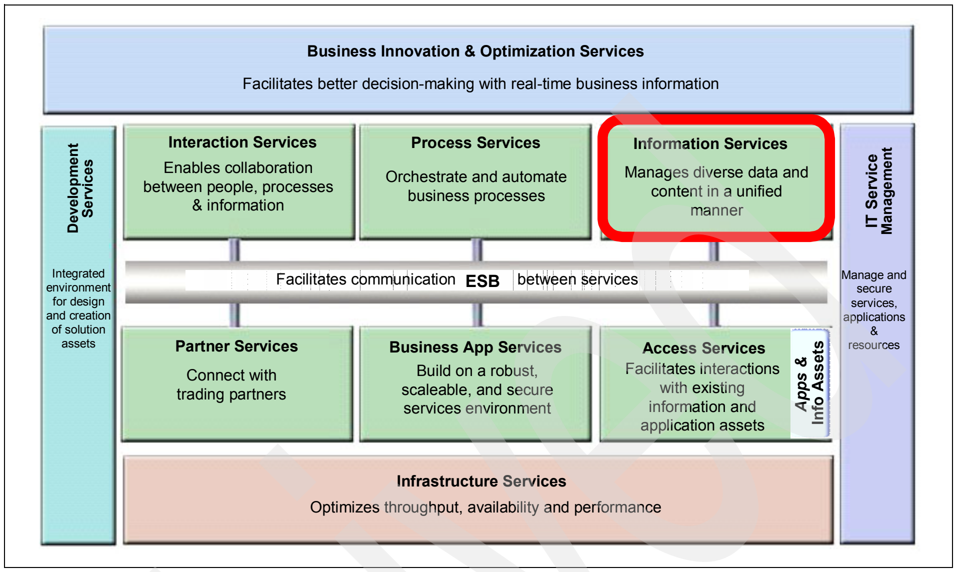

Business Innovation & Optimization Services
Facilitates better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enables collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manages diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
Facilitates communication ESB between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitates interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimizes throughput, availability and performance

Figure 6-2 SOA Reference Architecture and the scope of Information services

Information Services are the implementation in IBM's vision of how information should be open (through industry standards) and easily accessible (whenever needed, and freed from silos) to people, applications and business processes across the enterprise. This allows for improved agility, where information is no longer tied to proprietary systems, formats, or technologies. Organizations can now avoid vendor lock-in to a single database, operating system, or server platform.

Figure 6-3 on page 271 shows this new view.

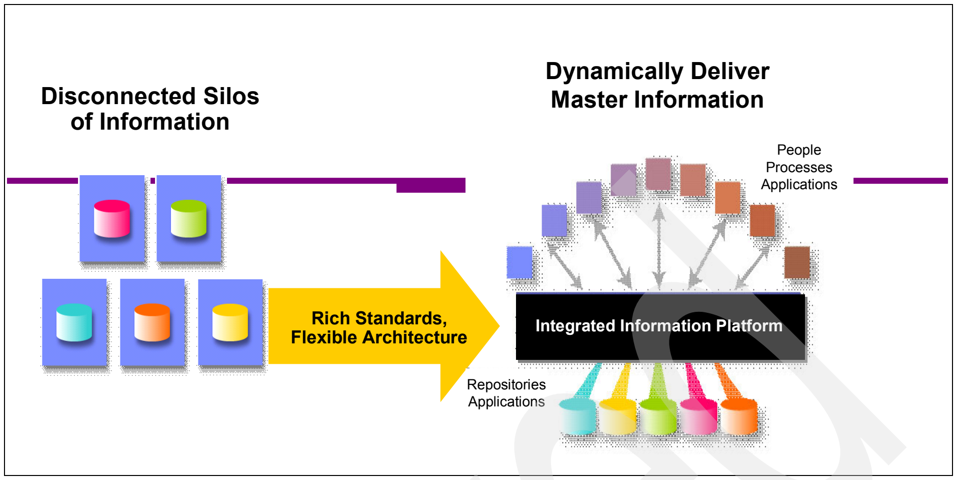

Disconnected Silos of Information
Dynamically Deliver Master Information
People
Processes
Applications
Rich Standards, Flexible Architecture
Integrated Information Platform
Repositories
Applications

Figure 6-3 Evolution of information architecture

Information services simplify development and maintenance of applications. This enables developers to easily connect to information without having to understand the specific Application Programming Interfaces (APIs) and semantics of those systems. Consistency of data processing and quality is maintained by the services that provide the data. Application developers focus on the design of new processes and applications, knowing that they are getting the best possible data from the services.

This idea of *Information virtualization* is depicted in Figure 6-4.

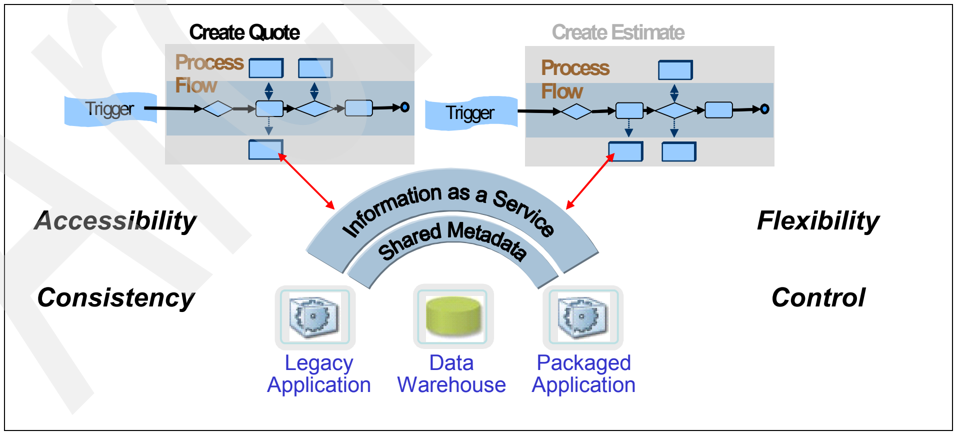


Figure 6-4 Information virtualization

6.1.3 Functions

Information services entails multiple areas of concern, as shown in Figure 6-5. Information services is also a part of the *Information On Demand* vision.

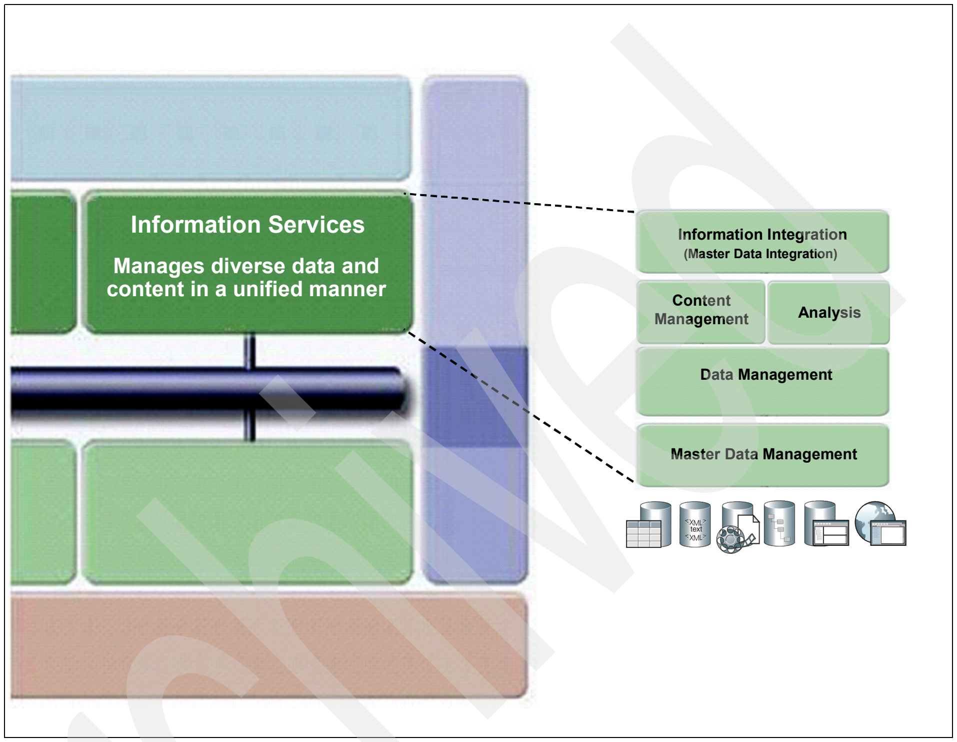

Information Services
Manages diverse data and content in a unified manner
Information Integration (Master Data Integration)
Content Management
Analysis
Data Management
Master Data Management

Figure 6-5 Information services components

Information services can be split up according to the components they implement:

- Data Management services
- Content Management services
- Analysis services
- Information Integration services
- Master Data Management services

We discuss these components in the following sections.

Data Management services

With a wealth of corporate data being on the mainframe, and a lot of it being locked in information silos, and within hard-coded applications, it is quite important and useful to unlock this information and make it available across the enterprise via platform-independent services.

The main functions of *Data Management services* are:

- Insulating applications from source changes
- Providing simple service enablement of data
- Supporting native XML and relational hybrid database
- Persisting messages and preserving integrity

A key opportunity is to manage XML data centrally. A lot of the inter- and intra-enterprise information exchange is now occurring via XML, and industry standards like HL7 for Health care, RFID for Retail, ACORD for Insurance, and SWIFT MX for Banking are popular. Of course the most common implementation of services in SOA is Web Services, and that typically relies on XML messages inside a SOAP envelope. The ability to store these messages in a central repository and the ability to analyze, summarize, transform, and then resurface the relevant portions as services becomes a key opportunity for Information as a Service.

Content Management services

Content Management services manage all kinds of data and content, access heterogeneous data, and publish data events.

They provide management of unstructured content through its complete life cycle, from capture of newly created content, access rights, storage policy enforcement, and publishing to enterprise policy enforcement and records management.

The main functions of Content Management services are:

- Providing easy service-oriented content access, so that you can access content easily from Portals, applications, and processes
- Applying service orientation for solving integration and management issues related to many different sources of content
- Providing simple and effective reuse of content across applications
- Providing heterogeneous content federation for accessing unstructured data from multiple content repositories
- Providing single service access to global content

Content Management Services surface a unified view across a variety of unstructured content like e-mails, images, documents, and so forth, so that this information can be leveraged throughout the corporation, including in the traditional database applications that typically work only on structured data.

Analysis services

Analysis services cover all processes, techniques, and tools that support business decision making based on information technology.

We use the term Analysis services to group *Business Intelligence (BI)* and *Data Warehousing (DW)* solutions because they are used interchangeably and the concepts depend on each other.

Typically BI includes end user tools for query, reporting, analysis, dash boarding and so forth. BI almost always assumes a Data Warehouse with timely, trusted information to be in place.

Both terms (DW and BI) address desire for timely, accurate, available data delivered when, where, and how the end users want it.

The major components of Analysis services are:

- Data store

  This data store contains the data used by the BI services and that is present in the data warehouse, the datamarts, and the operational data stores.
- Query and reporting

  The purpose of this component is to translate the business request into a database query and to present the result to the end user.
- *On-line Analytic Processing (OLAP)*

  OLAP is decision support software that allows users to analyze information making use of summarized multidimensional views and hierarchies. OLAP tools are used to perform trend analysis on information and enable users to drill down into masses of statistics.
- *Data mining*

  Data mining occurs when large quantities of data are analyzed and explored in order to discover hidden or unknown patterns that can aid decision making.

Information Integration services

Information Integration services allow real-time, integrated access to business information regardless of location or format, and provide the broadest set of information integration capabilities to align information across disparate sources based on its meaning.

The main functions of Information Integration services are:

- Harmonizing and virtualizing the information
- Analyzing, cleansing, transforming, federating, and publishing
- Delivering integrated, trusted information as a service

Data Integration provides an integrated view of data from multiple sources either using federation for real-time integration or using a consolidated data store.

Information Integration services combine data from a variety of data sources in an efficient manner, and with consistency, accuracy, and trustworthiness.

Master Data Management services

A very common opportunity that we observe for Information Services in an SOA context is *Master Data Management (MDM)*, which essentially provides a single authoritative source of information and a management interface for critical enterprise entities such as Customers, Products, and the like.

Figure 6-6 compares a situation without MDM with a situation with MDM.

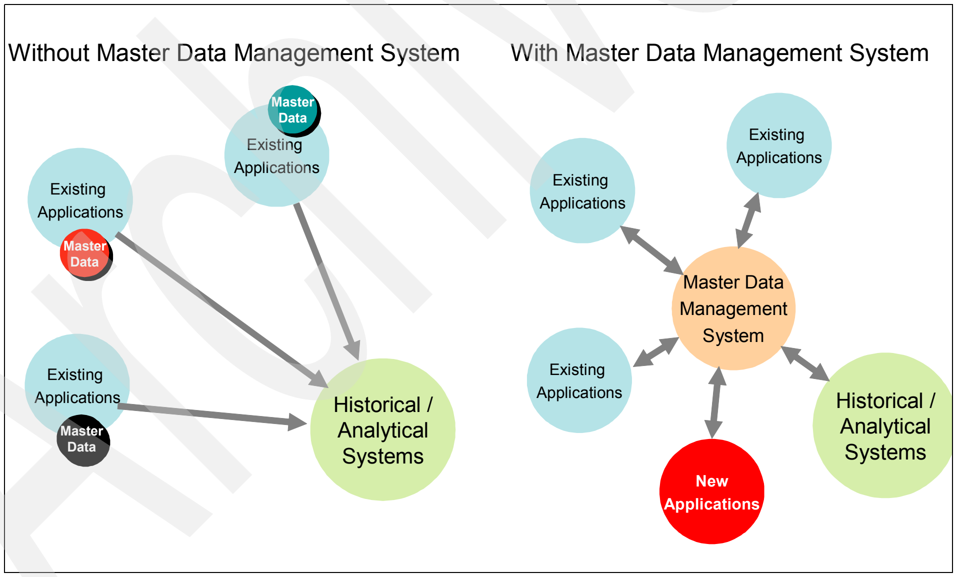


Figure 6-6 Principles of master data management

The fundamental principle of MDM is that master data is de-coupled from operational, transactional, and analytical systems into a centralized independent repository or hub. This centralized information is then provided to SOA business services so data is managed independently of any single line of business, system, or application.

This strategy enables enterprises to identify common functionality for all systems and applications and then support efficient, consistent use of business information and processes.

A Master Data Management strategy must acknowledge that certain information domains, such as product, customer, employee, and partner information, are relevant to most enterprise applications. And because each system potentially has its own database and varying levels of detail, queries to different systems might produce different results. This situation can cause costly data redundancies and misleading analytics.

Master Data Management is a set of disciplines, technologies, and solutions that you can use to create and maintain consistent, complete, contextual, and accurate master data for all stakeholders. Master Data Management uses the concept of master data objects, which represent the key business entities (for example, products, customers, and accounts) that an organization interacts with to run its business.

Master Data Management means defining what data is commonly used by many applications in order to define a new repository that can act as the reference for this information.

Master Data Management has many benefits:

- It enables the de-coupling of master information from individual applications.
- It simplifies the ongoing integration tasks and new application development.
- It ensures consistent master information across transactional and analytical systems.
- It addresses key issues such as data quality and consistency.
- It addresses these key issues proactively rather than “after the fact” in the data warehouse.

Master data is critical because it defines the business context for a particular domain.

6.2 Key architectural decisions with respect to System z

In this section we describe a checklist of key architectural decisions to be made for Information services with respect to System z.

Before making any of these decisions you have to create a complete overview of functional and non-functional requirements for the component you have to implement. You also have to keep in mind that in a lot of cases you will need to unlock existing data stores for new users. Once a list is created, the architectural questions identified in this section have to be answered.

The following questions concern the data store:

- Where is the business data that you have to access currently stored?
- Is it possible to relocate the data without impacting other users of this data?
- Is it possible to consolidate your data or is federation a better solution?
- Does the current data store meet your high availability requirement?
- How long does it take to recover the data in the case of a hardware failure or manipulation error?

There are also questions more specific to the Information services:

- What are the application access patterns?

  Do we have to deal with CPU-intensive calculations, OLTP, data warehouse, typical database, I/O bound, or data-intensive batch workloads?
- Is there an obvious fit for where Information services are hosted according to the location of the existing information data assets?
- Is your solution scalable to keep up with the expected growth?
- How will you deal with the availability requirements?
- Will the performance and response time stay acceptable?
- How will you protect confidential data?
- Will you be able to meet the security requirements?
- What about auditing requirements (including SOX, Basel II, and so forth)?
- Can the hardware platform deliver the required Quality of Service?
- Can the operating system deliver the required Quality of Service?
- Do hardware or software solutions exist that meet the Quality of Service (clustering, HADR, and so forth)?
- What skills and staff resources are available to manage the platform and does your implementation affect the number of required resources?
- What is the additional operational effort needed for monitoring, operating, and problem determination?

6.3 Common patterns

In this section we describe common runtime location patterns that will help us to identify the platform where the Information services should be located when the data store in on the z/OS platform. We do this independent of the implemented component.

Later we discuss the particular solutions for each component.

6.3.1 Information services on distributed

Even when the data store was located on z/OS historically, a lot of Information services-like solutions where placed on the distributed platform. See Figure 6-7.

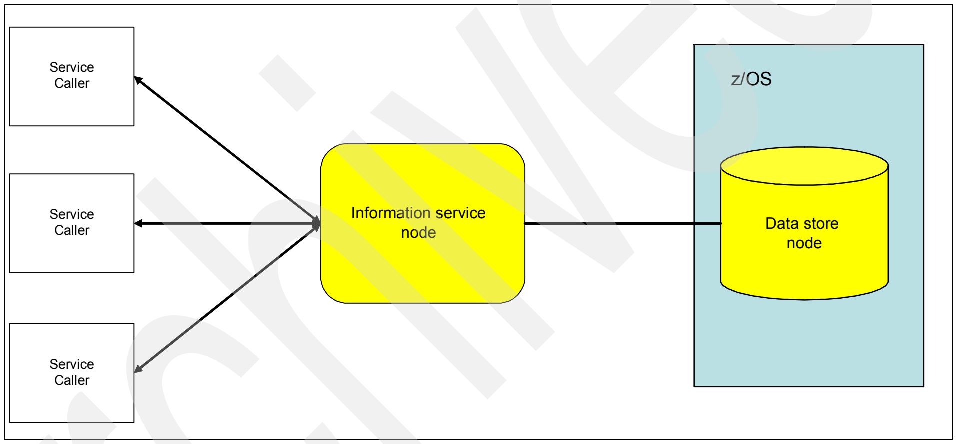


Figure 6-7 Base pattern for Information services on distributed

If you meet the end user requirements and you do not expect any scalability problems in the future the distributed platform can be an acceptable solution. Also, when there are no solutions (meaning ISV solutions) available for the implementation of your Information services on System z, this is sometimes the only possibility.

But once the enterprise starts with the transformation of its core business functions into new architecture models like SOA, the Quality of Service requirements increase. The requirements for some isolated applications for accessing Information services might not be so critical, but once your business becomes dependent on these Information services you probably want to have the same Quality of Service as you were accustomed to from your traditional online CICS and IMS applications.

If you have high data traffic or you cannot afford any latency you probably need to have your Information services very close to your data store to avoid potential congestion problems or delays.

Security and data privacy to address regulatory compliance can become a reason not to use this pattern.

Manageability and problem determination with this pattern is more complicated because you have to deal with different platforms and hubs that are interconnected via a physical network.

The pattern in Figure 6-8 shows a solution where the Information services access both data stores on z/OS and distributed and the previously discussed strengths and weaknesses apply.

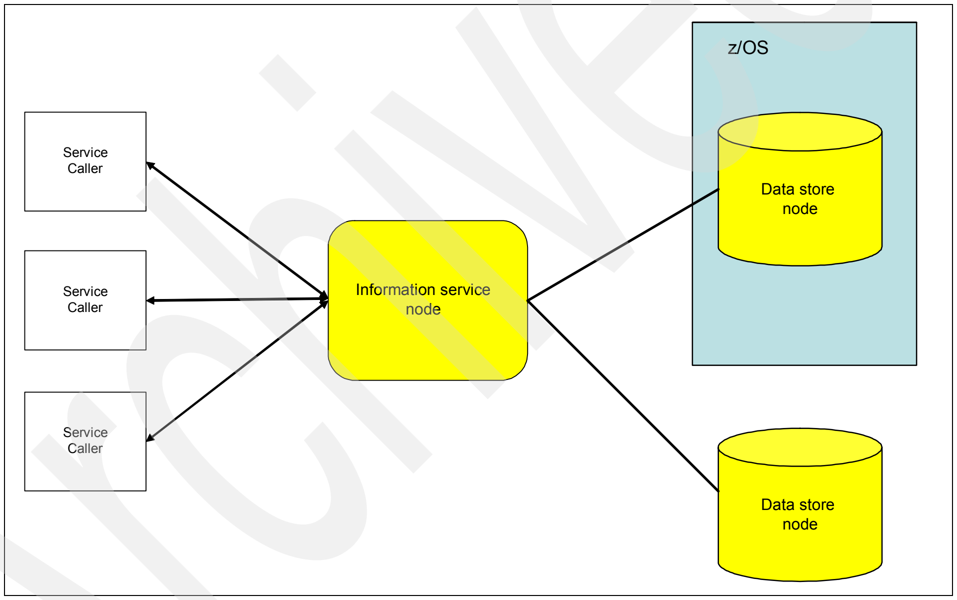


Figure 6-8 Pattern for Information services on distributed with mixed data stores

6.3.2 Information services on Linux on System z

In this pattern the data is stored on z/OS, but accessed by Linux on System z, as shown in Figure 6-9.

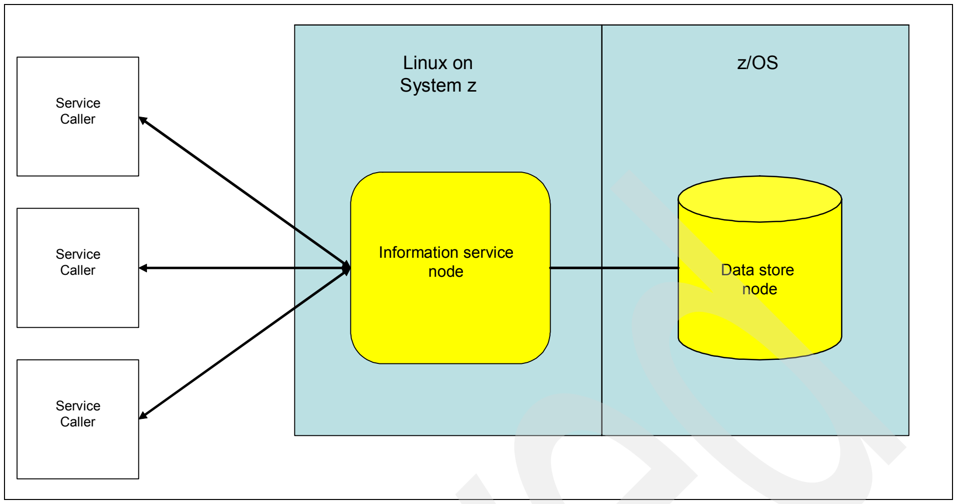


Figure 6-9 Base pattern for Information services on Linux on System z

When there are no solutions available for the implementation of your Information services on z/OS, you can consider using Linux on System z to bypass most of the restrictions mentioned in 6.3.1, “Information services on distributed” on page 278.

The Quality of Service is higher because you can make use of the reliability, availability, and security features of the System z hardware. But, if the business requires that you keep the Quality of Service level as high as it was with traditional mainframe applications, the z/OS operating system also contains some benefits that you will not find in the Linux on System z environment.

The data throughput between the Linux on System z systems and the z/OS partitions can be high because you can make use of *Hipersockets*. With Hipersockets you bypass most of the typical network overhead when communicating between TCP/IP nodes. However, the I/O throughput criteria can be so extreme that the only option is to place the data store on the same image as the Information services to meet the response time criteria.

If you have to deal with very stringent security and privacy requirements, placing the Information services on Linux on System z might not be adequate.

With this pattern, although you still have to deal with two different operating systems, problem determination and manageability becomes easier because you only have one hardware environment

Figure 6-10 shows a similar pattern where the data stores are on both z/OS and Linux on System z.

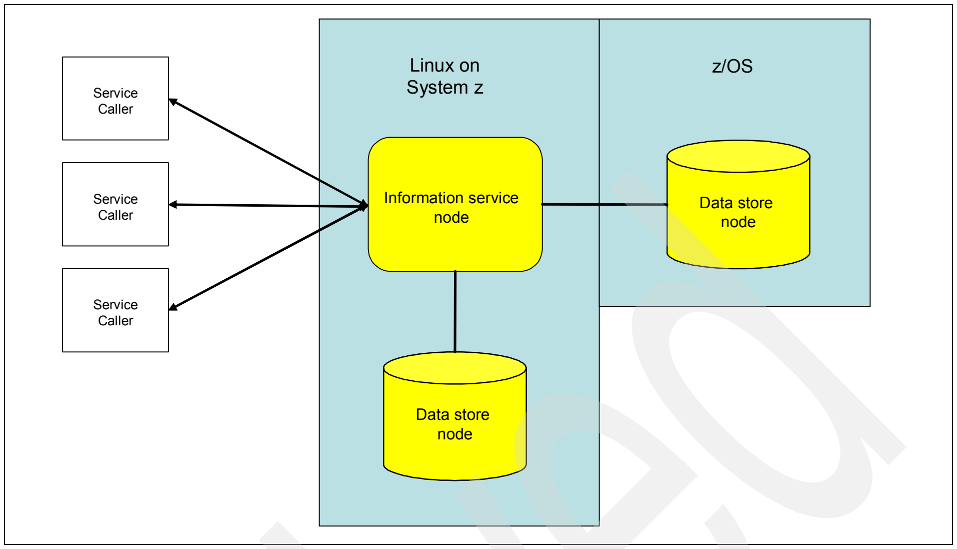


Figure 6-10 Pattern for Information services on Linux on System z

6.3.3 Information services on z/OS

In Figure 6-11 the Information services are implemented on the same system where the data store is located.

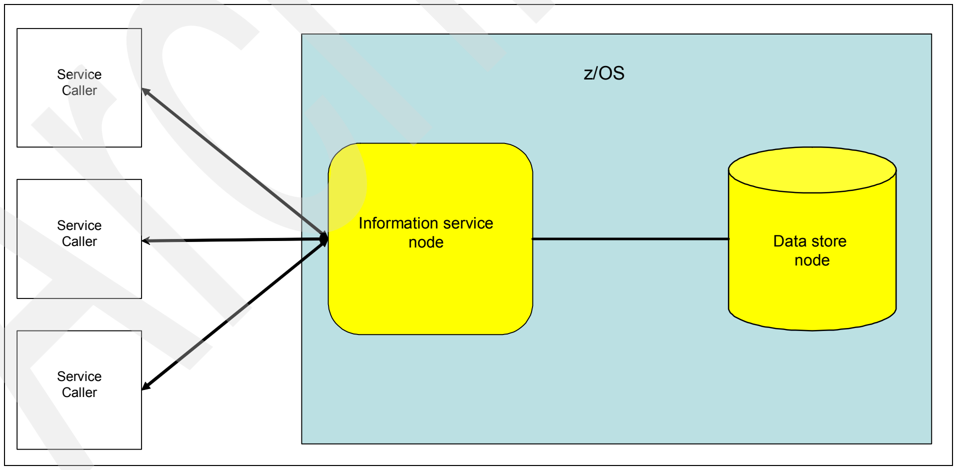


Figure 6-11 Base pattern for Information Services on z/OS

If you want to keep the same high reliability and availability as for your traditional mainframe applications the z/OS platform is the only solution.

The Information services should be on z/OS also when there is a need for strong security and data privacy to address regulatory compliance.

Furthermore, in cases where you cannot afford any I/O latency or you have to deal with high data traffic and want to avoid network traffic congestion problems, co-locating the Information services and the data store on z/OS might be the only solution.

6.3.4 Service callers on System z

In the following patterns the Information services callers are on System z.

Information services callers on Linux on System z

Once your service callers are residing on Linux on System z and your data is on z/OS it is useful to put your Information services on the same platform for performance, manageability, security, and scalability reasons. Figure 6-12 on page 282 shows two sub-patterns.

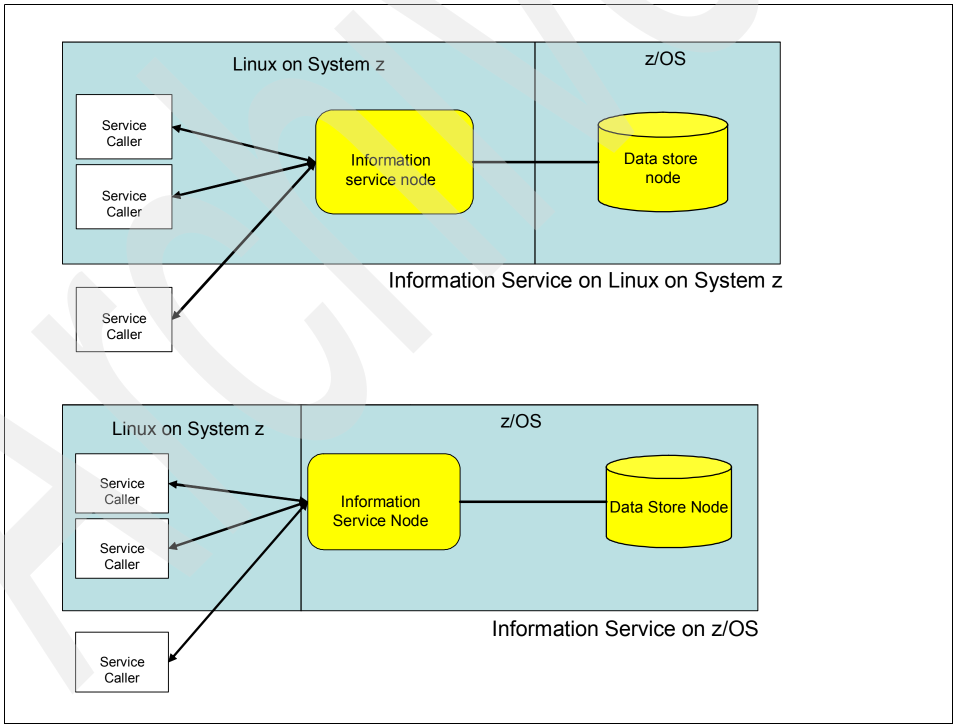


Figure 6-12 Pattern with service callers on Linux on System z

Service callers on z/OS

If your service callers and your data is on z/OS it is recommended to place your Information services on the same platform (Figure 6-13 on page 283).

This solution increases the benefits that we already stated, and adds new ones.

It gives you the opportunity to make the most efficient use of the z/OS resources you are allowed to use.

This solution delivers the best security and auditability possibilities.

Having an integrated solution improves the problem determination and diagnosis process and helps to quickly solve production issues.

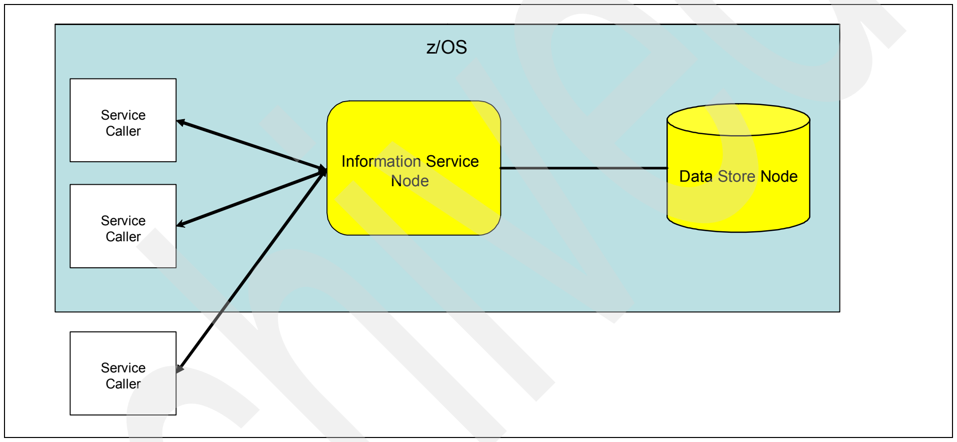


Figure 6-13 Pattern with service callers on z/OS

6.4 Platform choice criteria

In the following sections we discuss the platform criteria related to Information services.

6.4.1 Functional requirements

In a first phase you have to collect all the functional requirements. These requirements will depend on which components you are planning to implement. Data Management, Analysis, Content Management, Information Integration, or Master Data Management solutions fulfill different needs and all these components have their own typical functional requirements. Also potential future requirements have to be taken into account.

With rapidly changing business scenarios solutions must be flexible enough to manage demand for new data sources and new application requirements. With the complexity of available data, it has become difficult to predict which information is or will become relevant to users. New solutions must be built on an open standards-based architecture that integrates with IT infrastructures using industry standards (J2EE and XML).

A business might require a single view of customer data, irrespective of organization, while another, for business or legal reasons, for example, might insist that customer data is owned at a divisional level, and no single view of customer data is allowed. A data store which is large (and probably growing), and which also needs to support a high rate of read/write activity, will pose a very different design challenge to one which, although very large, is relatively stable and used only as reference (read only) data.

Each data store represents a subset of business data with different characteristics (that is, types based on volume, volatility, form), different usage, access, and management requirements. The characteristics used will generally be chosen based on the identified needs of the enterprise or the target business system.

The following are examples of those characteristics:

- Size
  How big is the data group - large volumes of static data versus transactional data?
- Rates of growth
  For example, 10% per annum versus doubled over the next 4 years?
- Volatility
  What is the rate of change of the data in the data group? 50% of the data changes weekly versus 10% of the data changes yearly?
- Currency
  How up to date does the data have to be? A maximum of one minute versus value at close of monthly trading?
- Form
  Is it text, image, structured, and so forth?
- Access
  Is the data mostly accessed “read only” or does it require update?
- Service characteristics
  Performance, availability, and so forth.
- Partitioning requirements
- Sharing requirements

- Management requirements

For a Data Management solution it will be important that it can deal with XML data because most SOA environments depend on information stored in XML messages.

A business analytics solution requires the generation of various forms of reports, charts, graphs, cube views, dashboards, and so forth, to enable users to make quick business decisions. Depending of your specific business requirements, a more or less sophisticated solution is needed. It must be possible to avoid expensive calculations and large amounts of I/O processing to create commonly used summaries and aggregates by keeping the results available for later use. It is the responsibility of the database optimizer to make use of these aggregations to accelerate queries.

For a Content Management solution it will be important to know what kind of content must be managed and where this existing content is located. Easy integration of various data sources to a DWH database is required. In today's environment, the data is derived from a lot of different sources. Besides DB2 databases on System z and on distributed systems, non DB2 databases, legacy sources on System z like IMS, VSAM, and multiple formats from distributed systems need to be accessed.

A Federation service must be able to interact with a range of different data structures types, including:

- Relational and hierarchical databases (DB2, IMS, Adabas, CA-IDMS, and so forth)
- Various mainframe file formats (VSAM, QSAM, HFS, and so forth)
- XML
- Packaged applications for Customer Relationship Management (CRM) and Supply Chain Management (SCM)
- Semi-structured data, such as Web sites, collaboration tools, content repositories, imaging and workflow systems

6.4.2 Security

Security is one of the most important non-functional requirements for data.

Data stores hold data that contains confidential information like customer details, business processes, business strengths, and so forth. This data has to be kept secured to avoid any intrusion. It also must be possible to encrypt the data. Security needs to be enforced so that only individuals with a "need to know" can access specific data and query the appropriate information without other data being compromised. Also, databases must provide the ability to implement row-level security under the control of a central security system to limit data access only to information the user is untitled to.

The authorization system needs to provide fine-grained access control over virtual repository nodes and other application objects. On the mainframe, in the past the data store and transaction servers (such as CICS and IMS) resided on one platform without any external network connections and protected by a central security system (RACF or Top-Secret). Now, these traditional data stores need to be available to the SOA landscape world were flexibility and loose-coupling are the keywords.

For new Information services, it is important to integrate with the existing security infrastructure and to leverage, where possible, existing authentication and authorization information stores. This would avoid the need to create, populate, and maintain a separate authorization or entitlement store just for new information services.

Sensitive data should not be stored in a readable format either in the active database or in the back-up copies on cartridges or other external media, so this data must be encrypted.

A Single Sign-on system can be used to authenticate and manage users in a multi-user, multi-repository application environment, and to store and retrieve the credentials used for logging into various content and workflow repositories. Applications can use the Single Sign-on System to provide users with seamless access to content stored and managed in disparate back-end systems without requiring multiple logins.

6.4.3 Auditing

In recent years companies have been required to comply with new regulations such as BASEL II, Sarbanes Oxly (SOX) and the US Patriot Act to improve financial reporting, avoid fraud, and protect investors' interests. Regulations also emphasize the need to reproduce sequential versions of data and business states, and keep auditing trails for a long period of time. Fewer data sources reduce the audit management effort, so it would be useful to consolidate business data in one platform.

A few other examples:

- Sarbanes Oxley (SOX) requires companies to keep more data than ever before. The combination of our environments growing and the need to keep the data longer can cause a data explosion.

- The Health Insurance Portability and Accountability Act (HIPAA) regulations force companies to ensure all sensitive data is appropriately protected. HIPAA also demands tight security of all sensitive data.

6.4.4 Information latency

There is a growing need to understand and react quickly to the immediate concerns of the business.

This means that queries in a BI or Content Management environment need access to near real-time data. The OLTP data changes should be replicated to a DWH database with no or small latency. To enable ad hoc queries that require real-time information, all data needs to be available to the analytical users all the time without the need to request and to wait for it. The DWH data is initially extracted from the OLTP environment and ongoing updates are required. The performance impact on OLTP must be minimized.

The process of extracting information from the OLTP system by necessity runs on the same platform as the OLTP system. The process of transforming it to conform to the needs of the target system is also usually performed on the platform where the OLTP system resides. This is because many of the techniques used in the transformation process reduce the amount of data that then must be moved to and loaded into the target system.

There are many methods that can be employed in this movement of data. Whatever method is employed, if the target location is physically distinct from the platform housing the source system, it is necessary to transmit this data over a communications path of some kind. Very often, this is the single most expensive component of the entire ETL process. Obviously, if the target application resides on the same platform as the source system, sending data over a communication path is unnecessary.

6.4.5 Data store and access

The environment must be able to keep up with the increasing amounts of data it must handle. The increasing number of users and services adds additional storage requirements.

High I/O delays from the data storage devices can lead to performance problems for transformation processes, which sometimes need to read huge amounts of data from the same volume. The infrastructure needs to be able to serve multiple I/O operations to the same logical volume to reduce device queue times.

Data compression must be possible so that it takes less disk space to store the enormous amounts of data. Compression and decompression must be handled with no additional overhead, if possible.

The data store must deliver support for XML data by combining the management of structured and unstructured data to allow you to store XML data without any overhead.

Data-intensive workloads like large databases and transaction processing work that have I/O rates and synchronization needs run better in a large shared resource SMP architecture with shared work queues and caching. Workloads that rely on processor power will most likely perform better on distributed server engines.

6.4.6 Existing landscape

Few businesses ever have the luxury of being able to start their information services from scratch. Chances are high that your business has been up and running for a while, and furthermore your business has probably already been exploiting information systems to store business data. There is a very good chance that you already have applications that implement function that fits into your business design.

Also your corporate organization will play a role. If your organization tends to work in a centralized way, centralized keeping or consolidating of your database management on System z will be very useful. If, on the other hand, your business units operate in a very decentralized manner, it can make more sense to put your DBMS on distributed platforms.

6.4.7 Availability

In the past most production databases had to be available during “normal” business hours, but globalization and the Internet revolution have forced many businesses to now run 24*7 without any downtime. So continuous availability of these databases has become essential.

Even Data WareHouses in a BI solution are moving from the back office to the front, so that BI solutions require the same availability as OLTP systems. A DWH environment should be available 24*7. Downtime due to database changes, or OS or software upgrades are no longer acceptable.

This means that database outages must be transparent to the user. Clustering of systems and databases and automatic takeover must be implemented.

6.4.8 Scalability

Reuse plays an important role in SOA. As a consequence, if your solution is well designed, in the future other consumers will use your Information service. Of course this increases the need for scalability because you can hardly predict the future usage.

When planning for the workloads that a data server must support, a systems architect has two primary options for “scaling,” or providing the required hardware processing capacity and resources to a data server.

The first approach is to “scale up,” where capacity is increased by adding processors and memory to a single physical machine up to the maximum capability of the hardware. With this approach, the data server cannot share data processing tasks with other separate systems. This means that once the database workload has exceeded the limitations of this system’s hardware, it must be replaced with hardware of higher capacity.

The second approach is to “scale out,” where a cluster of data servers executing on multiple separate hardware systems operate together in a coordinated manner to serve data from the same logical database. Any “node” in the cluster can then service any incoming client request for shared data, and for resiliency clients can connect to any operational node remaining in the cluster during a failure.

6.4.9 Resilience and recovery

For critical services any hardware or software failure must be transparent to the service by the implementation of high availability solutions. This means doubling all the hardware and software components and the implementation of an architecture such that failure of one or more of the components, or a complete disaster, will not interrupt services for the end users.

Backup of the data must be possible with little or no disruption to the services. It must also be possible to transparently switch to secondary disks that contain a copy of your data, to protect to both planned and unplanned re-configurations.

Because of the emerging needs for BI and DWH solutions, they have the same requirements for backup and recovery of the data as online applications.

6.4.10 Maintainability

For critical services you have to be able to execute “rolling” maintenance and upgrades. This means that the maintenance and upgrades of the data stores must be executed without any service interruption for the end user.

6.4.11 Manageability

It is important to design your solution so that it in the long term stays manageable. Consolidation of data stores makes it possible to manage the same amount of business data with less staff and reduces the daily management so that the data administrators can spend time on tasks that add more value for the business.

6.4.12 Performance

Because applications with different SLAs are accessing the same data stores it is very important to have the possibility to prioritize work. There is a need for a workload manager that is able to monitor the performance goals and make the appropriate decisions when needed. This means assigning more resources to critical services and delaying or blocking less important work. This capability is very critical in a consolidated environment where multiple types of workload run on the same hardware and compete for the same resources. It also must be possible to view whether the performance goals are reached.

Another aspect of performance is response times. Co-location of the Information services and the data store increases the throughput because fewer network hops means less time on a wire.

6.4.13 Accounting

For charge-back purposes the platform should be able to report the resource consumption (CPU, disk space) to the application owners and end users.

You must be able to aggregate this information and put it in a data warehouse for performance reporting and capacity planning purposes.

6.5 Data Management services

In this section, we discuss a number of common patterns as they relate to Data Management services.

6.5.1 Distributed application serving and data consolidated on z/OS

In this option, the application servers are running in a separate environment and they access the data on the System z platform (Figure 6-14).This environment allows for data consolidation on the System z platform (z/OS) and results in numerous benefits.

However, there are some trade-offs:

- The connections between the application servers and the System z platforms are made with external network protocols.
- Two different teams with different sets of skills are needed to operate this environment: the distributed team and the System z team.

Traditionally, using System z data capabilities would have required processing power of the system. Because some software licenses are based on the total capacity of the machine, it might have increased the cost of putting data on the System z platform. This can be addressed with IBM offerings such as the *System z Integrated Information Processor (zIIP)* specialty engine and enhancements to DB2 for z/OS middleware, which are designed to help customers integrate data across the enterprise, improve resource optimization, and lower the Total Cost of Ownership (TCO) for data-serving workloads.

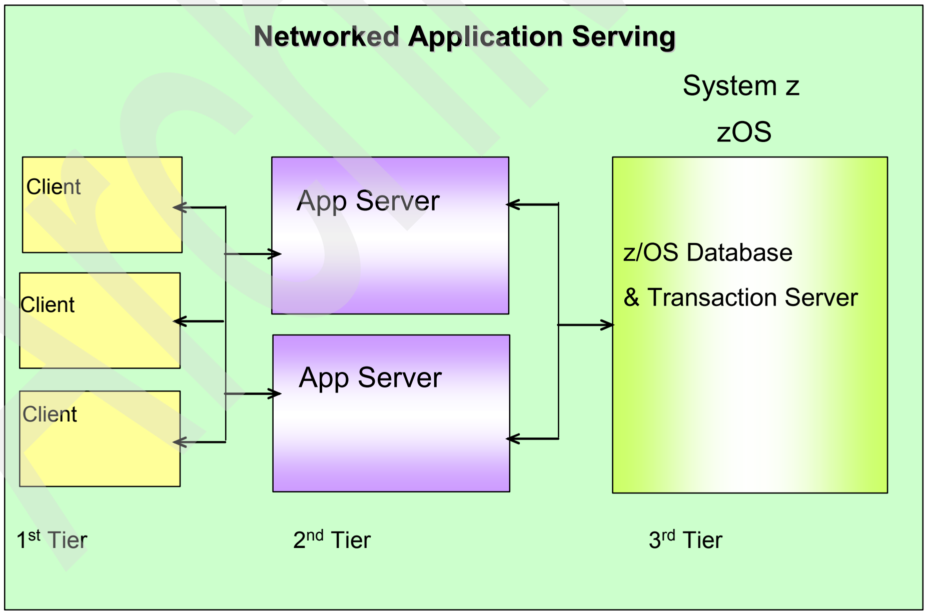


Figure 6-14 Distributed application serving and data on z/OS

Distributed application serving consolidated on System z

With Linux on System z you can consolidate the application servers in Linux on System z partitions on the System z platform (Figure 6-15). Communication between partitions of the System z hardware is done through a very efficient mechanism known as HiperSockets. HiperSockets eliminate the need to utilize I/O subsystem operations and the need to traverse an external network connection to communicate between logical partitions in the same System z server.

For this feature the operations team must be knowledgeable in both Linux and z/OS. It is also interesting for Independent Software Vendors (ISVs) who can leverage the portability of Linux applications while benefiting from the strengths of the System z hardware platform as an enterprise hub for data.

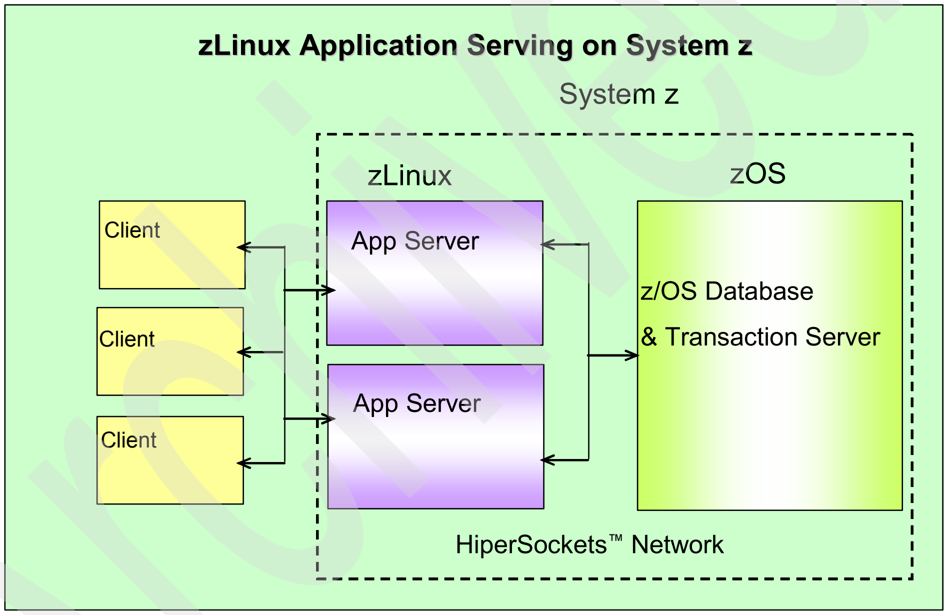


Figure 6-15 Distributed application serving consolidated on System z and data store on z/OS

Integrated application serving and data store on z/OS

The applications that run directly on the System z platform under z/OS will leverage the benefits of locating applications and data in the same technical environment (Figure 6-16). Communications between the application servers and the database manager will use a very efficient cross-memory mechanism.

The operations team will have a single environment to manage.

The classic applications written in languages such as COBOL or PL/I will run within a classic z/OS transaction manager such as CICS or IMS. The new applications can also be written in Java and execute within CICS or IMS as well, or benefit from WebSphere Application Server for z/OS, a certified J2EE application server running on the System z platform.

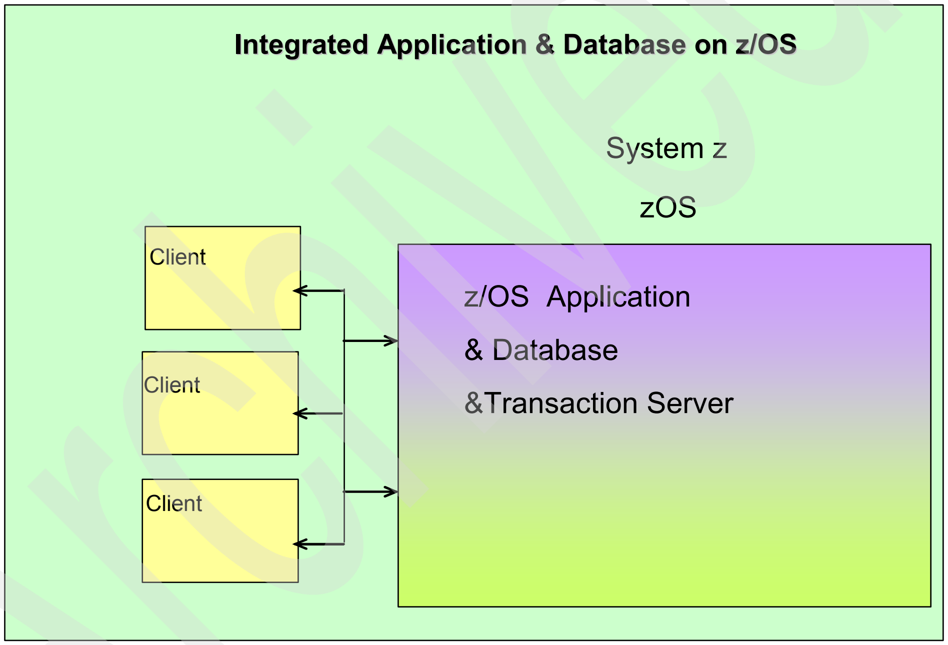


Figure 6-16 Integrated application serving on z/OS and data store on z/OS

Data consolidation on the System z platform

Many organizations have a lot of applications and the data that those applications manipulate is scattered in many places. Data consolidation on the System z platform without having to change the applications can bring many benefits (Figure 6-17).

Data consolidation on the System z platform helps reduce:

- Number of data copies, and hence the risk of disparate data
- Cost and complexity of backup and recovery
- Network traffic
- Amount of storage needed through centralization and efficient hardware data compression
- Database administration and management tasks
- Risk associated with distributed privacy, security, and audit policies

With data consolidation, customers leverage System z technology through:

- Use of Parallel Sysplex clustering for scalability, availability, and performance
- Data-sharing capabilities that allow them to get a single view of data
- Centralized backup, recovery, privacy, security, and audit policies

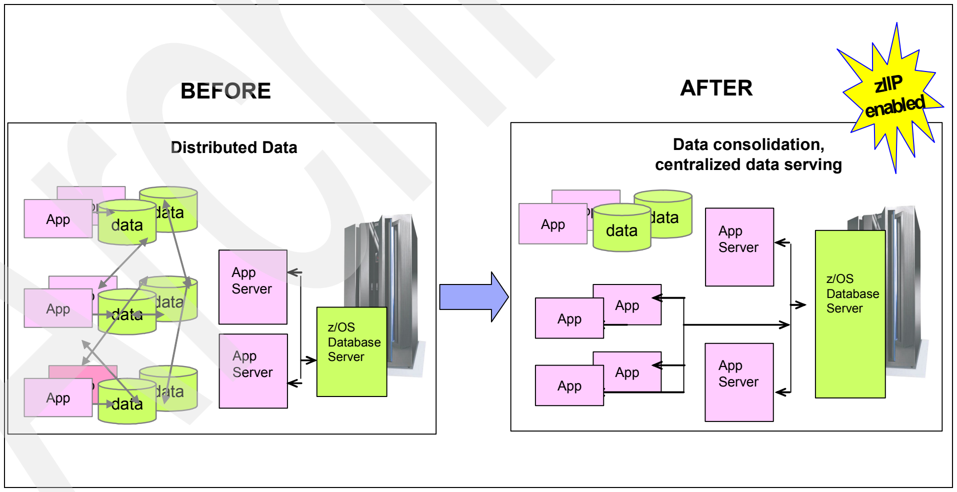


Figure 6-17 Data consolidation on System z

Data consolidation and application integration on System z

In addition to data consolidation, it is possible to integrate some applications on the System z platform with Linux on System z (Figure 6-18).

With the integration of applications closer to the data, more benefits are available, in particular:

- There are fewer points of intrusion in a simpler network so security is improved.
- The resilience of end-to-end operations is enhanced because there are fewer points of failure. Continuous availability is easier to put in place because of the characteristics and capabilities of the System z platform.
- The portion of response time due to network latency is reduced because data and applications are located on the same platform and communicate though the very efficient HiperSockets.
- Operations team efforts are simplified because there are fewer parts to manage.
- The overall solution uses less hardware, so power and cooling needs are reduced, which improves the environmental impact of the solution.
- Specialty processors such as the IBM Integrated Linux Facility (IFL) and the System z9 Integrated Information Processor (zIIP) offer those benefits at an attractive cost.

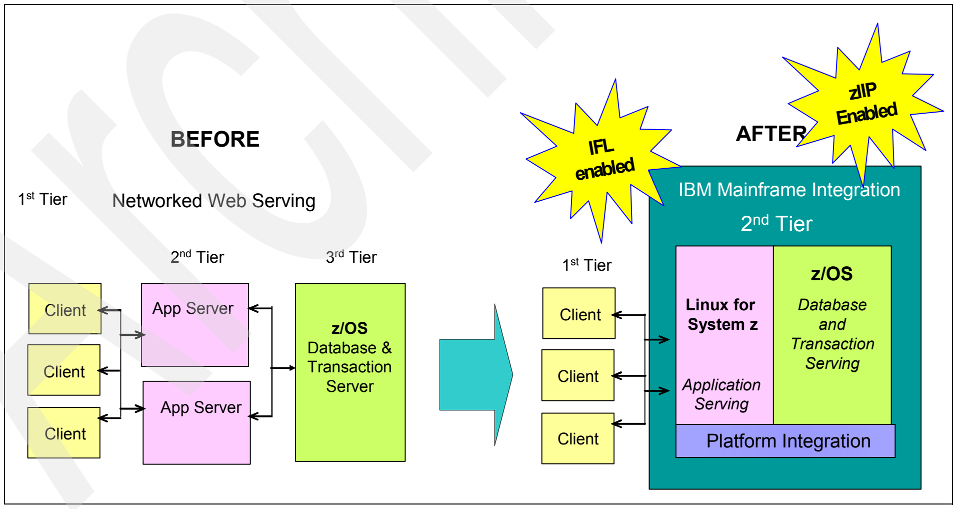


Figure 6-18 Data consolidation and application integration on System z

Data consolidation and integration of applications on z/OS

From a technical point of view, the solution that brings the most value to the enterprise is having the data consolidated on the z/OS environment with the applications running there as well.

This solution is obvious for customers who are already running applications on the z/OS platform and need to extend them. It represents a good move in other cases, where enterprises can benefit from the portability of J2EE distributed applications to WebSphere Application Server on z/OS.

This solution provides all the benefits that we stated previously, and adds new ones:

- In this environment, the management of identities is more consistent, and the solution enhances auditability.
- The z/OS system is optimized for efficient use of the resources it is allowed to use.
- Transaction processing and batch work can be done at the same time on the same data, which improves availability and versatility.
- If an issue occurs, the integrated problem determination and diagnosis tools quickly help solve it.
- Automatic recovery and rollback ensure a superior level of transactional integrity.
- The Java workload created by J2EE applications can benefit from the specialty processor System z Application Assist Processor (zAAP).

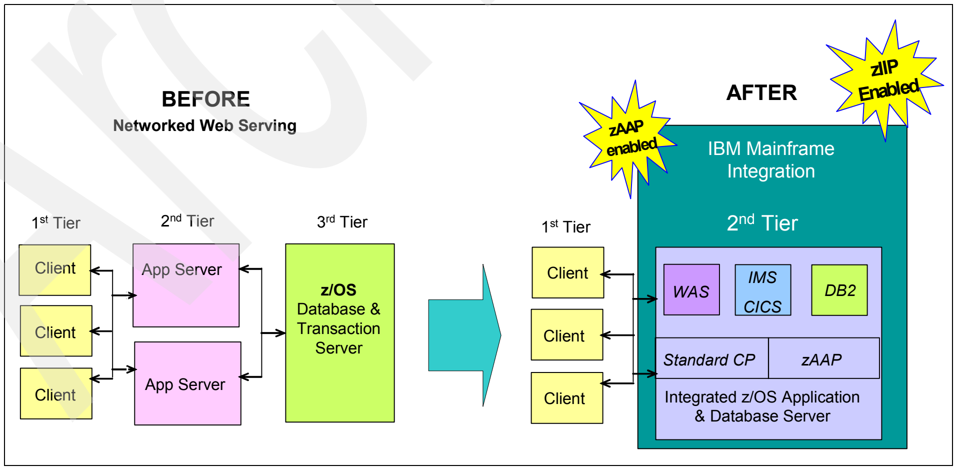


Figure 6-19 Data consolidation and application integration on z/OS

6.5.2 Data stores on System z

The System z platform has several file or database managers, some of which are particular to z/OS. In the following sections we discuss the most common ones and how they can contribute to an SOA effort.

DB2 on z/OS

Without a doubt, DB2 on z/OS is the leading Relational Database Management System for the System z platform. The IBM System z platform and DB2 for z/OS have continually set the standards that other systems attempt to reach.

The requirements of mission-critical environments can best be achieved through deep integration of the data server with the hardware, operating system, middleware, and tools. This is also true for DB2 on z/OS, where the full value of DB2 is achieved through an optimized combination of hardware, operating system, APIs, and tools. Probably the best example of this is DB2 data sharing leveraging the System z Parallel Sysplex implementation.

Scalability

Due to the unique hardware clustering that System z provides in Parallel Sysplex, DB2 for z/OS scales “linearly” (the addition of more hardware produces a corresponding proportional increase in throughput). It scales well beyond a few nodes and does not hit a throughput saturation level, meaning it does not suffer from a “glass ceiling” of performance like most other DBMSs.

DB2’s tight integration with z/OS and System z hardware facilities improved performance and scalability. DB2 exploits System z hardware features like built-in encryption, compression, and so forth. This results in reduced CPU consumption and better application performance. Most other DBMSs are designed generically to run the same code on multiple operating systems and use software to accomplish functions like compression and encryption. Using software to do these functions might degrade performance.

DB2 for z/OS is capable of “scaling up” and “scaling out.”

A single System z10™ is capable of scaling up to 64 processors (over 30,360 MIPS of processing power) and 1.5 TB of main memory. To prevent outages, a standby system is typically put in place and kept synchronized through various mechanisms, thereby staying ready for failover in the event of planned or unplanned downtime on the primary data server.

For scaling out DB2 for z/OS provides a feature known as *data sharing,* which integrates with and leverages Parallel Sysplex clustering technology and the System z hardware environment.

DB2 for z/OS data sharing technology leverages the unique capabilities of System z hardware and Parallel Sysplex technology to allow multiple different DB2 subsystems to read from, and write to, the same DB2 data concurrently. These different DB2 subsystems constitute what is known as a DB2 *data sharing group*. The subsystems can be spread across multiple separate System z hardware units within the cluster, but are presented to applications as one single database instance. At the core of this technology is a specialized processor known as the *Coupling Facility (CF),* which provides high-speed caching and locking functions. This enables multiple members of a DB2 data sharing group to read and update the same data simultaneously. DB2 for z/OS utilizes these functions to enable data sharing in two main ways. First, each individual DB2 subsystem has access to shared DB2 caches within the CF called the *Group Buffer Pools*. These Group Buffer Pools provide shared caches from which any member DB2 subsystem can quickly access data without requiring disk I/O. Second, to coordinate simultaneous access to shared data, DB2 utilizes specialized resource locking functions provided by the coupling facility. Most customers now use the *Internal Coupling Facility (ICF)* which uses “link-less” connections for the z/OS LPARs that reside on the same box. This allows transmitting shared cache and lock data at machine speed (clock cycle speed for that particular processor). External CFs are connected with dedicated high-speed fiber optic links (FICON® Express4) of 6 GB per second.

It is very important to understand that these global locking mechanisms and the synchronization state or “coherency” of the group buffer pools are directly managed by the coupling facility hardware. This means that individual DB2 subsystems do not expend their processing resources maintaining the state of shared cache data and locks. Also, because individual DB2 subsystems communicate with the coupling facility for shared cache and lock data and not with one another, there is very little increase in communication overhead as the number of member subsystems increases. This is a unique benefit offered by the coupling facility hardware and is a key reason why DB2 for z/OS data sharing exhibits near linear scalability as additional DB2 subsystems are added to the data sharing group.

IBM has measured the overhead of adding additional DB2 subsystems to a data sharing group and has shown that each additional member in a DB2 data sharing group adds a minor incremental performance overhead of not more than 1%. To provide redundancy, multiple coupling facilities can be configured within the same Parallel Sysplex and the cache and lock data stored within them can be “duplexed” or synchronized so that no cache or lock data will be lost in the unlikely event of a hardware failure.

When the System z architecture was extended in the mid 1990s, an additional instruction was implemented called *Send Message (SMSG)*. This instruction allows LPARS and the Coupling Facility to communicate with machine speeds (in the order of microseconds) without the overhead of a TCP/IP-like communication protocol. This SMSG instruction is embedded within the *Cross Memory Services (XMS)* API used to communicate with the Coupling Facility. At machine speeds with current technology, it is possible to get 100,000+ communications with CF per second.

Another impressive achievement of the Coupling Facility is the ability to write and read from other node buffer pools *without* interrupting the General Purpose processor. This is called “invalidating the buffer” and is done by using special hardware to turn on a “dirty bit” maintained by DB2 alongside the data. This mechanism allows the main processor to continue working and eliminates the need to “context switch” across the main processors. Invalidating buffers is done in a broadcast fashion in parallel with all nodes in a data sharing group at machine speed.

Thus, DB2 for z/OS within a Parallel Sysplex can scale throughput linearly without the additional messaging overhead that is required for distributed architectures in the absence of a centralized hardware cache.

As with DB2 for z/OS data sharing, scaling out distributed DBMSs must also provide global locking and shared cache management functions. In most cases these functions must be implemented in software through a complex message passing algorithm where all nodes in the cluster constantly communicate with each other to coordinate the state of the distributed locks and shared cache. Unlike DB2 for z/OS that benefits from centralized lock and cache management within the System z coupling facility, the distributed locking and cache management of most distributed DBMSs requires significant inter-node communication and processing resources in each node to service lock and cache requests from the other nodes, thus reducing transaction throughput. As additional nodes are added to the cluster, and as the amount of data sharing increases, the amount of inter-node messaging and processing overhead increases while the scale-out benefits from adding additional hardware decrease.

To alleviate these negative performance effects, administrators of distributed DBMSs must reduce the amount of inter-node communication occurring in their system by limiting the extent to which multiple nodes share the same data. Known as “partitioning,” this can require multiple application and schema changes. Not only can this be very difficult to accomplish, especially for a highly variable OLTP workload such as an ERP application, but it also results in down time when data is no longer accessible to the application. Partitioning essentially eliminates the advantage of the shared data scale out architecture because the emphasis is to configure the system to avoid sharing data. In addition, partitioning makes the process of adding an additional node more complex because the database partitioning scheme must be modified.

The proof of such scaling is demonstrated in a recent IBM internal benchmark performed for a core banking application for the Bank of China. They required 4,000 transactions per second to combine separate databases into one common database.

DB2 over-achieved the original target and drove the throughput to 9,445 transactions per second. This was accomplished with just a 4 node cluster. Since then IBM has run another benchmark for an Asian bank surpassing 15,000 transactions per second with a 4 node cluster. The limit of Sysplex technology throughput has yet to be discovered.

For a press release regarding this benchmark, refer to:

```
http://www-03.ibm.com/press/us/en/pressrelease/21044.wss
```

To make the best use of system resources, to maintain the highest possible throughput, and to achieve the best possible system responsiveness, System z provides the z/OS Workload Manager (WLM). WLM, an integral part of z/OS, allows you to define performance goals and assign a business importance to each goal. WLM then ensures these goals are met by monitoring the system to determine how much resource, such as CPU and storage, should be given to a particular workload to meet its processing goal.

Applications running on a mainframe benefit from the virtualization capabilities in the Processor Resource/System Manager (PR/SM) and Intelligent Resource Director (IRD) functions.

In summary, because of IBM's unique hardware clustering, synergy with System z architecture, exploitation of embedded hardware compression and cryptography, and integration with z/OS Workload Manager, DB2 for z/OS is the superior architecture base for an enterprise data server.

Availability

The design philosophy of DB2 is to deeply integrate with, and exploit all the core strengths of System z hardware and the z/OS operating system. This minimizes planned outages because hardware and software maintenance and common administrative activities can be performed during regular application operation with no operational down time.

IBM Parallel Sysplex technology is designed to keep the application and subsystem state information within the coupling facility. This enables a "rolling window" update approach that allows for nodes and applications to work seamlessly across the cluster if one node goes down. Most DBMSs typically require all nodes to be at identical software levels and need down time for maintenance. Even when rolling window updates are possible, in most cases availability is affected during the maintenance load of patches.

DB2 for z/OS Version 9 provides the ability to modify tables while the database is running, to change system procedures on the fly, to reorganize and load on line, and to execute fast, system-level point-in-time recoveries.

IBM takes database availability to the highest level possible with DB2's support of GDPS disaster recovery and continuous availability solutions. DB2's participation in GDPS assures customers true full data availability (24x7x365) for Data Serving on DB2 for z/OS.

Resilience

With DB2 for z/OS, unplanned outages due to hardware or software failure are highly unlikely because the System z environment is inherently resilient to hardware and software failures and can recover from or adjust to most problems transparently without affecting availability or response times. However, if a DB2 subsystem is lost or taken offline due to hardware or software failure, existing connections and new incoming requests are automatically routed to another available member of the DB2 data sharing group, keeping the data available to the application. The Coupling Facility prevents any loss of cache or locks, and there is no impact to running queries. Recovery of the locks held by in-flight transactions on the failed node occurs when the failed DB2 subsystem is automatically restarted and, with the exception of data pages locked by the failed DB2 instance, all shared data is continuously available to the other DB2 group members. DB2 allows for graceful workload transition in case of a node failure. All major database connecting applications, like DB2 Connect™, CICSPlex®, WebSphere Application Server, and SAP exploit this. In case of a node failure the workload is recovered on surviving members and no new workload is sent to the failed node. Thus any hardware or software failure is transparent to the application.

Security and compliance

The cost of security breaches can be significant for today's enterprises. IBM System z has a centralized security model, which has been tested and proven to be the most secure system for over 40 years in the most demanding of enterprises. DB2 for z/OS has needed only 9 security patches in over 20 years.

DB2 provides security mechanisms such as native DB2 authorization (grant/revoke), DB2 secondary authorizations, and DB2/RACF interface.

System z servers also provide end-to-end hardware encryption. Tools like Audit Expert, zSecure, and Optim™ Data Privacy provide audit, alert, and archiving capability to secure enterprises real time and provide reporting for compliance purposes. Optim tools ensure data privacy over the life cycle of data with policies and rules for creation, retention, archiving, and purging at a business user level.

Maintainability

z/OS fully supports the DB2 "data sharing environment." One advantage of the DB2 for z/OS data sharing environment for applications is that DB2 software maintenance and upgrades can be performed in a rolling fashion that has minimal if any impact on running DB2 applications. Rolling upgrades allow DB2 data sharing group members to be taken off line and upgraded separately, while other DB2 members remain on line and available. This high availability feature is possible because DB2 subsystems within the same data sharing group can simultaneously share data even while they are running different levels of DB2 software, or even different z hardware levels. Once application servers have been redirected, a DB2 member subsystem can undergo maintenance with no impact to connected end users because incoming work processes are routed to active DB2 data sharing group members.

The most significant and obvious limitation of most distributed DBMSs is that database upgrades are not supported by this rolling upgrade capability mechanism.

Integration

IBM is attuned to the latest needs of application programmers and the Independent Software Vendor (ISV) community, and works closely with all application vendors to develop and deliver key features, rather than competing as an application provider. In the last two releases of DB2 9, over 90 features were shipped based on ISV requirements from partners like SAP. Such features include optimizer enhancements to benefit data warehouse applications like SAP's Business Warehouse. IBM works very closely with SAP and certifies DB2 with new SAP releases within 30 days.

DB2 for z/OS 9 provides *pureXML®*, a native XML storage technology, providing hybrid relational and XML storage capabilities (Figure 6-20 on page 303). This provides a huge performance improvement for XML applications while eliminating the need to "shred" XML into traditional relational tables or to store XML as Binary Large Objects (BLOBs), methods other vendors use.

DB2 9 for z/OS presents a plethora of innovative functions and features. At the top of the list is the newly integrated DB2 XML storage engine that supports pureXML. It gives you the ability to store and query your XML data in its inherent hierarchical format.

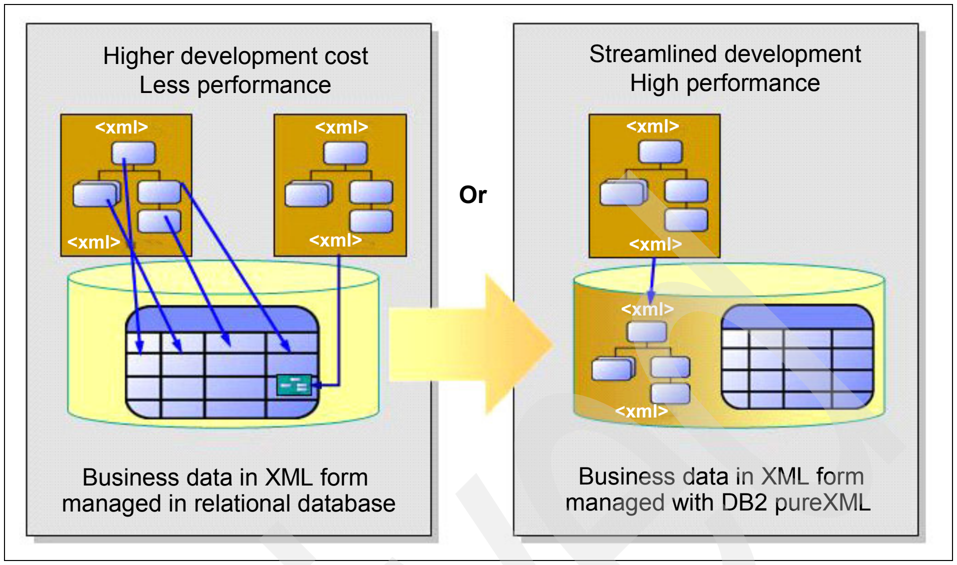
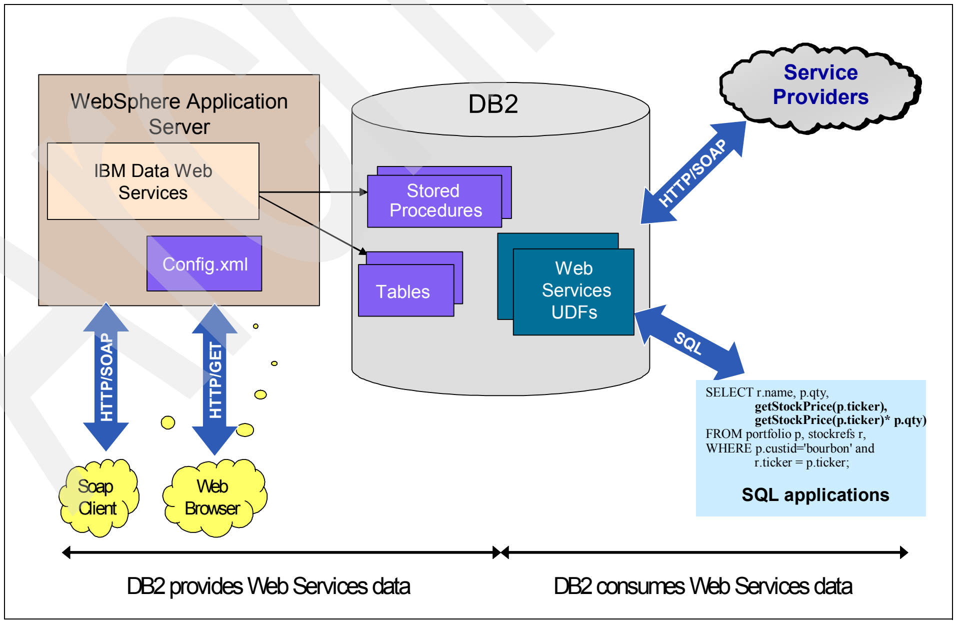


Figure 6-20 DB2 V9 - A new generation hybrid data server

Also new is the IBM *Data Web Services (DWS)*, which allows you to build and generate Web services based on DB2 data and stored procedures. Figure 6-21 shows an overview of this technology.



```
SELECT r.name, p.qty,
        getStockPrice(p.ticker),
        getStockPrice(p.ticker)* p.qty)
FROM portfolio p, stockrefs r,
WHERE p.custid='bourbon' and
        r.ticker = p.ticker;
```

Figure 6-21 Web Services support in DB2 on z/OS

IMS Database Manager

Information Management System (IMS) can play an important role in SOA implementations. One of the main advantages of IMS is that it has managed hierarchical data for almost 40 years now.

Because XML and IMS databases are both hierarchical, IMS is a natural DBMS for managing XML documents. IMS will allow easy retrieval and storage of incoming XML documents. It can compose XML documents from existing legacy information and store them in its databases.

The IMS database server is deeply integrated with the System z hardware, the z/OS operating system, other mainframe middleware, and tools.

Scalability

IMS has scalability through virtualization to assure flexibility for growth and expansion. This works in a homogeneous environment utilizing many heterogeneous data and application resources with new hardware and software facilities to optimize performance, capacity, availability, and recovery. This includes new levels of enhanced availability for IMS *High Availability Large Databases (HALDB)* with fully integrated online reorganization support that provides concurrent online update and availability of data. The maximum size of an HALDB database is 4 gigabytes (data set size) x 1001 partitions x 10 data sets per partition or approximately 40 terabytes of data.

IMS provides increased capacity and incremental horizontal growth, offering improved availability with network, message and data sharing. It utilizes the Coupling Facility of the System z platform and the latest technological advancements for security and integrity of z/OS. IMS provides open/integrated access with Java and XML. IMS also provides improved systems management through automated operations, workload balancing, dynamic routing, dump analysis, and packaging enhancements.

IMS continues to strengthen its support of the enterprise by providing the highest in performance, availability, security, integrity, at the least cost per transaction.

IMS fully exploits the new technology and power of z/OS and the Parallel Sysplex. Existing IMS data sharing capability was initially enhanced to take advantage of the Coupling Facility for storing lock information and for easy availability of that information by all systems in the Sysplex environment. The lock manager in each system could access the locks as they needed to. In addition to data sharing, IMS provided necessary information to the WLM workload manager on z/OS to assist with workload balancing of resources across the Sysplex.

IMS also enhanced message routing between systems to take advantage of workload balancing information, and IBM provided the IMS Workload Router to use these facilities to push the work to the available system. Significant enhancements were also added to complement the Parallel Sysplex hardware and operating systems facilities. IMS has since improved its initial Data Sharing and Workload manager enhancements with additional data sharing (storing changes and unaltered data on the coupling facility for Sysplex access, and providing additional Fast Path sharing), message sharing (providing message queues and fast path messages on the coupling facility for Sysplex access), and message routing enhancements (utilizing VTAM® Generic resource support). As customer workload grows, the power of distributing data and applications across the Sysplex is needed. End users want to be able to access applications and data transparently, regardless of where the work is processing. This enhanced support provides improved end user interaction, improved IMS availability, improved workload balancing, and offers increased capacity and growth in moving into Parallel Sysplex environments.

Manageability

Implementing IMS as multiple instances or in a Parallel Sysplex provides support of a Single Point of Operations Control (SPOC), presenting a single system image and allowing the user to enter commands to all IMSs in the IMS Sysplex (the IMSPlex) from a single console. Although designed with Sysplex in mind in order to optimize operations across a Sysplex, the new SPOC can also be used to improve systems management of commands in general, and the SPOC can be used to control any IMS, without the requirements of a Sysplex. This support can provide operations management for IMS DB from a TSO/ISPF application running on z/OS or from a distributed DB2 with its IMS Control Center code.

Integration

IMS applications and data can use and be accessible using the latest in standard architectures and interfaces, including the J2EE Connector Architecture (JCA) APIs for transactional access of IMS programs and the Java Database Connectivity (JDBC) interface for database access.

IMS uses a hierarchical model as the basic method for storing data, which is a pragmatic way of storing the data and implementing the relationships between the various types of entities. In this model, the individual entity types are implemented as segments in a hierarchical structure. The hierarchical structure is determined by the designer of the database, based on the relationships between the entities and the access paths required by the applications.

IMS provides the ability to store and retrieve XML data natively. IMS converts non-XML data to XML for interchange and converts it back or stores it natively.

IMS V9 introduced a way to view your IMS data as collections of XML documents, aligning the stored IMS records with the valid XML documents for retrieval and automated conversion between these. The emerging *XQuery* standard is a powerful query language to search, aggregate, evaluate, pick and choose parts of the XML collection, and then convert the resulting data into XML. IMS XQuery support provides access using standard XQuery expressions to IMS data, including new IMS XML data. This opens up IMS to the emerging market of off-the-shelf third party tools and offers a standard shareable integration point between IMS and other industry databases (Figure 6-22).

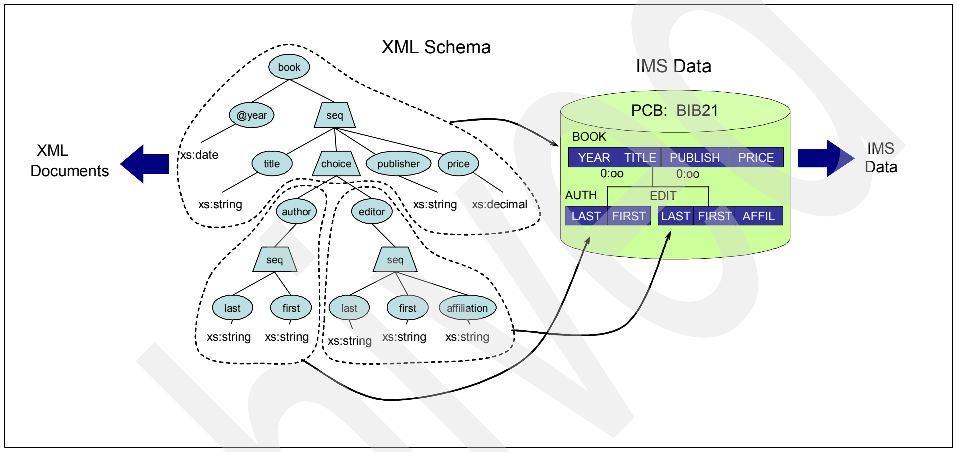


Figure 6-22 IMS as XML database

6.6 Content Management services solutions

In the following sections we discuss the Content Management solutions for System z.

6.6.1 IBM Content Management solutions

There are many Content Management solutions available that can be implemented to access data on System z. The most significant IBM solutions in this space are listed here, along with Web addresses to consult for detailed specifications:

- IBM Information Integrator Content Edition
  (`http://www.ibm.com/software/data/integration/db2ii/editions_content.html`)

- IBM Content Manager OnDemand (`http://www.ibm.com/software/data/ondemand/`). This product is available for i5/OS®, z/OS and Multiplatforms
- IBM Omnifind (`http://www.ibm.com/software/data/enterprise-search/omnifind-enterprise/`)
- IBM Filenet Content Manager (`http://www.ibm.com/software/data/content-management/filenet-content-manager/`)

6.6.2 Content Manager for z/OS v8.4

IBM Content Manager for z/OS (CM z/OS) manages all types of content (document images, electronic office documents, XML, audio, video, and so forth), with support for a broad range of platforms, databases, and applications. It was developed from the ground up with a multi-tier, XML-ready data model, including Java, XML, and Web Services programming interfaces.

If a large part of the document management environment is located on IBM System z it makes sense to also put the Content Manager on System z to benefit from data proximity advantages and the QoS of System z such as high reliability, availability, and security.

Figure 6-23 shows the architectural components of CM z/OS.

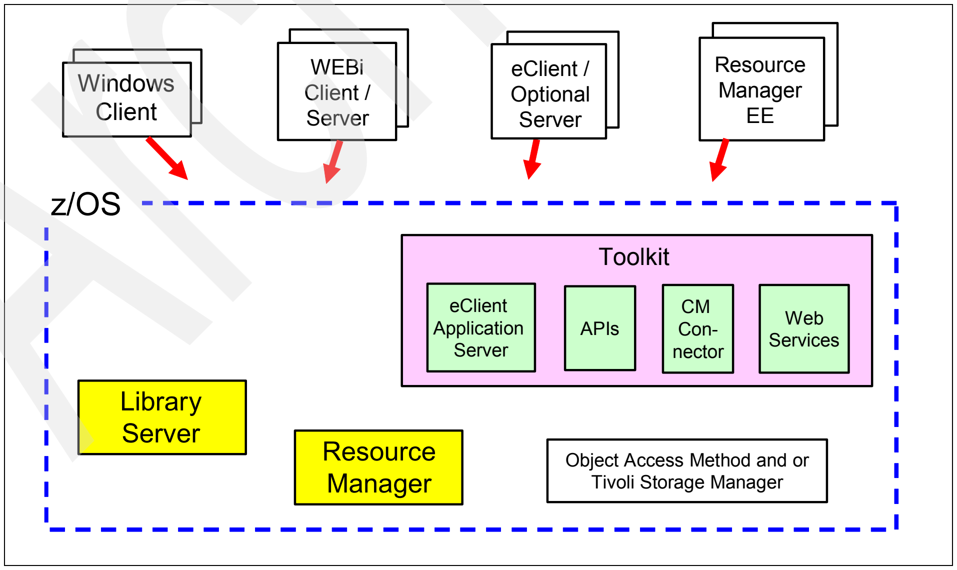


Figure 6-23 Content Manager z/OS - Architecture components

The main components are:

- The *Library Server (LS),* where the searchable index data (metadata) is located.
- The *Resource Manager (RM)* acts like a traffic cop and does not store the actual objects. RM calls APIs for the *Object Access Method (OAM)* or *Tivoli Storage Manager (TSM)* and it is OAM, TSM, or both that actually control the physical storage of the objects. RM performance is critical and for this reason RM is implemented using Common Gateway Interface (CGI).
- The *Toolkit* also includes Web Services support.
- The clients to access the servers. The eClient Application Server is implemented in WebSphere Application Server on z/OS or WebSphere Application Server on Linux on System z and can make use of the QoS delivered by WebSphere Application Server on System z.

Figure 6-24 displays the implementation of CM z/OS.

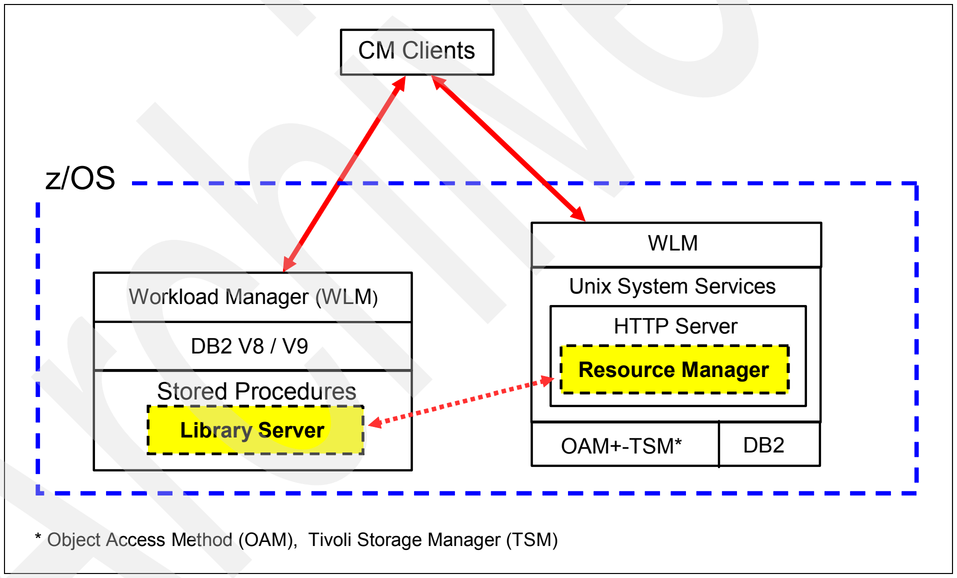


Figure 6-24 Content manager z/OS implementation

The Library Server is implemented as more than 100 DB2 stored procedures and Workload Manager is exploited for managing DB2 under which the LS is executing.

The Resource Manager stores objects in OAM or TSM, or both. OAM is a DB2 application that manages and tracks the objects. The objects can be stored on DASD, tape, or optical.

RM is implemented as a CGI application for performance reasons versus Java in the Enterprise Edition RM and runs under the control of the IBM HTTP Server in UNIX System Services. Workload Manager is exploited for managing the HTTP Server where RM is executing.

For high availability (HA) and disaster recovery (DR) the eClient Application Server can be deployed in WebSphere Application Server on Linux on System z or WebSphere Application Server for z/OS.

Figure 6-25 shows the CM z/OS client architecture.

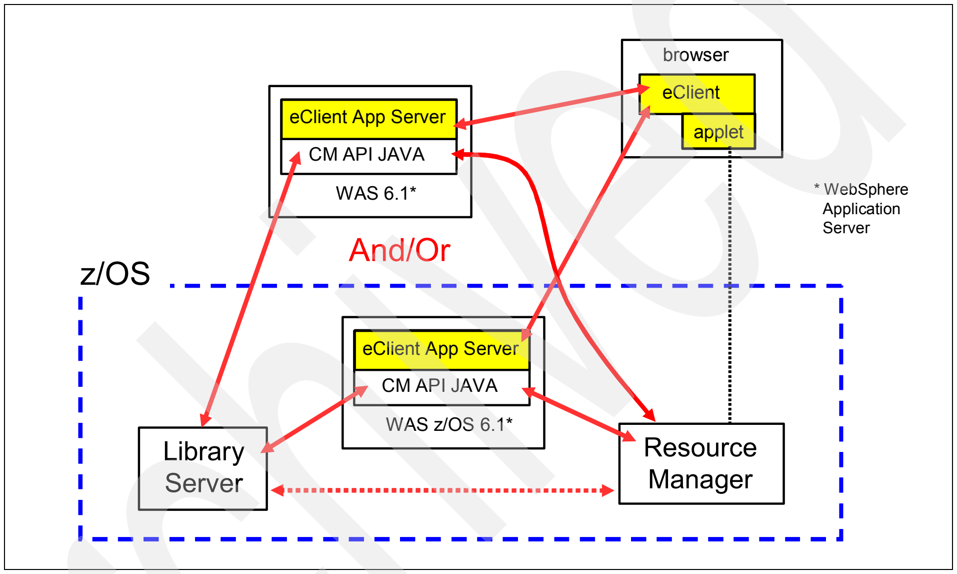


Figure 6-25 Content Manager z/OS - client architecture

Content Manager z/OS can be implemented in an HA setup using System z Parallel Sysplex, as shown in Figure 6-26 on page 310.

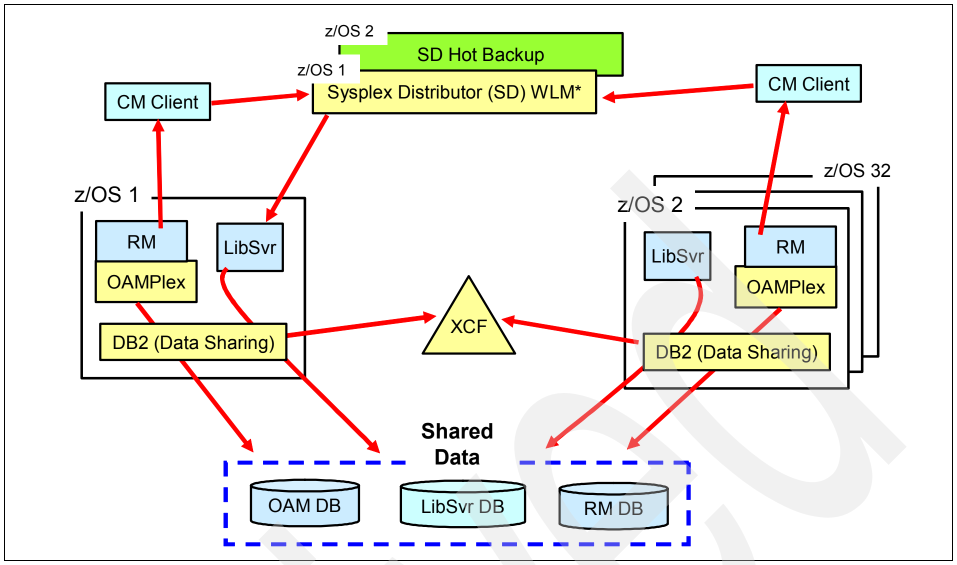


Figure 6-26 Content Manager z/OS in Parallel Sysplex

A CM z/OS solution on a single System z provides a high availability and high capacity solution. Parallel Sysplex enhances the high availability of a System z application as well as increasing the capacity.

A Parallel Sysplex environment has a Sysplex Distributor (SD) and Workload Manager (WLM) which handle incoming requests from clients and direct the requests to an available back-end application based on the amount of workload. Think of this as a clustered solution providing a single image to the end user. There are no “hot standbys.” All systems are live.

Referring to Figure 6-26, z/OS 1 has a Library Server, Resource Manager, and an “OAMPlex” with DB2 Data Sharing implemented. DB2 Data Sharing allows DB2 applications on separate System z LPARs to access the same databases. The LS, RM, and OAMPlex can exploit DB2 Data Sharing if implemented. DB2 Data Sharing uses the Cross Coupling Facility (XCF) for enforcing integrity.

The CM Client sends in search requests to the Sysplex Distributor (SD), which routes the request to the LS in z/OS 1.

The LS performs the metadata search against the LS database and sends the results back to the client.

The CM Client chooses the object to be retrieved, the request goes to SD, then RM on z/OS 1, OAMPlex to retrieve the object from media, and RM then sends the object to the CM Client for rendering.

Now one can add additional System z LPARs running z/OS and participating in the same Sysplex, thereby increasing capacity and scalability.

This allows requests to and from the CM client to be routed to any z/OS system based on workload and availability. There is no data affinity. If the LS on z/OS 1 fails all requests go to the LS on the other z/OS LPARs dynamically. If a complete z/OS fails the other z/OS systems pick up in real time. If the Sysplex Distributor fails its hot backup takes over in real time. The databases are shared by all images using XCF to enforce data integrity.

6.6.3 Content Manager OnDemand for z/OS

Content Manager OnDemand (CM OnDemand) enables high-volume computer output, such as statements, invoices, and back office reports, to be captured and managed. It also supports archiving for e-mails, scanned documents, checks, and instant messages.

The solution can run on System z to provide a highly reliable system for enterprise report management and electronic statement presentation.

CM OnDemand is optimized to capture, search, present, and manage large collections of small objects such as statements or bills.

The core server and client components offer the following capabilities:

- Report and document capture handles multiple data types.
- Searching, viewing, printing, and faxing options are varied and easy to use, and include support for annotations and logical viewing.
- Hierarchical document storage management across magnetic, optical, and tape devices and facilities for moving data between storage devices and for managing backup and disaster recovery.
- Document viewing through Web browsers, portlets, and CICS systems.
- Built-in security features allow administrators to control access to the system and its archived documents.

The Web interface employs open standards and supports Web 2.0 and AJAX technologies.

OnDemand server and its components

The OnDemand server environment includes a library server and one or more object servers that reside on one or more nodes.

A CM OnDemand library server maintains a central database about the reports stored in CM OnDemand. The database manager provides the database engine and utilities to administer the database. The library server processes client logins, queries, and print requests, and updates to the database. The major functions that run on the library server are the request manager, the database manager, and the server print manager.

A CM OnDemand object server maintains documents on cache storage volumes and, optionally, works with an archive storage manager to maintain documents on archive media, such as optical and tape storage libraries. An object server loads data, retrieves documents, and expires documents. The major functions that run on an object server are the cache storage manager, CM OnDemand data loading and maintenance programs, and the archive storage manager.

The basic CM OnDemand configuration is a library server and an object server on the same physical system or node. This single library or object server configuration supports the database functions and cache storage on one system. You can add an archive storage manager to the single library or object server configuration to maintain documents on archive media.

In the standard configuration your library server and object server reside on the same node.

You can also configure your CM OnDemand system with a library server on one node and one or more object servers on different nodes. This configuration is known as a distributed library/object server system. The distributed library or object server configuration supports caching of documents on different servers. You can add an archive storage manager to one or more of the object servers to maintain documents on archive media attached to different servers.

Content Manager OnDemand for z/OS includes the capability to mix and match servers to support the architecture and geographic needs of the customers, while allowing the data to be kept closer to the requesting users, minimizing network traffic. For example, a company with a z/OS-based library server and object server can add an additional object server in another location that can be on any CM OnDemand platform. Or you can choose to run CM OnDemand library server on a distributed platform, with the object servers on z/OS.

Figure 6-27 shows a configuration of CM OnDemand on z/OS with high availability.

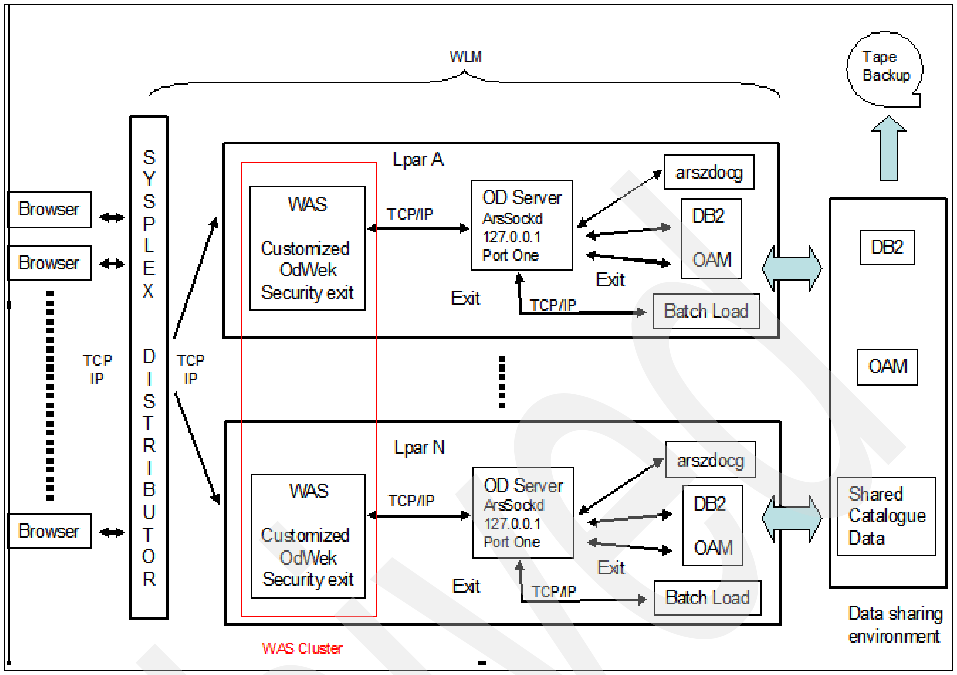


Figure 6-27 CM OnDemand configuration on z/OS with high availability

The system components include:

- A Sysplex Distributor for distributing incoming TCP/IP requests to one or multiple LPARs.
- Each LPAR contains a complete Content Manager OnDemand for z/OS WebSphere implementation. This includes WebSphere Application Server, containing user applications, the Content Manager, the Content Manager OnDemand server (ARSSOCKD), and V2 compatibility code (ARSZDOCG).
- Each of the LPARs connects to DB2 and OAM. Both DB2 and OAM are installed in a data sharing environment and act as the data archive backbone of the system. The data sharing environment allows Content Manager OnDemand index and object data stored from all of the LPARs to be placed in a single DB2 and OAM datastore. Thus, any single LPAR is independently capable of accessing all the OnDemand data and functioning independently of any other LPAR.

6.6.4 DB2 Content Manager Enterprise Edition

DB2 Content Manager Enterprise Edition manages all types of content, such as document images, electronic office documents, XML, audio, and video, all with support for a wide range of platforms, databases, and applications.

IBM DB2 Content Manager Enterprise Edition V8.4 adds platform support for Linux on System z. The library server and resource manager components, Java API and Content Manager (ICM) connector, Web services, XML API, and the eClient can now be installed and used on a 64-bit Linux on System z operating system with 64-bit DB2 databases.

DB2 Content Manager Enterprise Edition uses the security and auditability features of the host hardware or software.

IBM DB2 Content Manager Enterprise Edition provides universal access to content while preserving system performance, whether configured for a single installation or a distributed environment.

It delivers the repository services and infrastructure platform for IBM DB2 Content Management solutions such as production imaging, case management, compliance in a regulated life sciences environment, compliance with records management, document life cycle management, e-mail management for Lotus Notes® or Microsoft Exchange Server, digital media, and Web content management.

IBM DB2 Connect Manager Enterprise Edition offers an XML-ready data model, Java, C++, and Web Services programming interfaces, and an integrated hierarchical storage manager that supports hundreds of storage devices and media types.

The resource manager can make use of WebSphere Application Server ND clusters to increase availability and scale horizontally and vertically, with enterprise features such as replication to store and manage objects in multiple locations and application-transparent content caching.

6.7 Analysis services solutions

In this section we discuss several Analysis services solutions.

6.7.1 Common location patterns for the Data Warehouse

Two common situations with regard to placement of the Analysis services solution are covered in this section.

Distributed DWH solution

In this configuration, information hosted on System z, such as data extracted from DB2, IMS, VSAM, Adabas, and CA-IDMS, is offloaded into separate data warehouses that run on the distributed platform, such as UNIX and Windows. Until recently, that strategy remained standard industry practice. Warehousing on System z typically created additional operational overhead that reduced overall mainframe performance. Distributed warehousing platforms such as DB2 and Oracle also offered greater functionality. For this reason moving data off System z for reporting became very common.

At this moment some data is still better served from distributed data warehouses.

The most important reasons to choose a distributed DWH solution are:

- When the DWH is relatively small.
- When a hybrid solution (System z plus LUW) is not viable for high analytic/OLAP requirements or heavy data mining requirements (unless the customer wants to use SAS or SPSS for mining).
- When there is a clear direction to reduce mainframe workload, for example, with the intent to use more mainframe resources for master data or running transactions.

But for highly sensitive information, data that changes frequently, or data you need to access in real time, a mainframe-based strategy can bring better performance and security and reduced liability to the equation without adding additional cost.

Within a distributed data warehousing environment, the risk of an unauthorized person accessing or exposing crucial data continues to grow, with copies of data kept on multiple servers or in numerous repositories. The risk increases as companies grant systems access to new users.

It becomes more complex to document compliance across numerous distributed computing applications and systems to monitor and log system access and use because your data is stored on several locations.

Centralized DWH solution

For the reasons just discussed, a centralized DWH solution might be preferred. Figure 6-28 shows the concept of running a DWH on System z using two footprints (“CEC”) and DB2 Data Sharing.

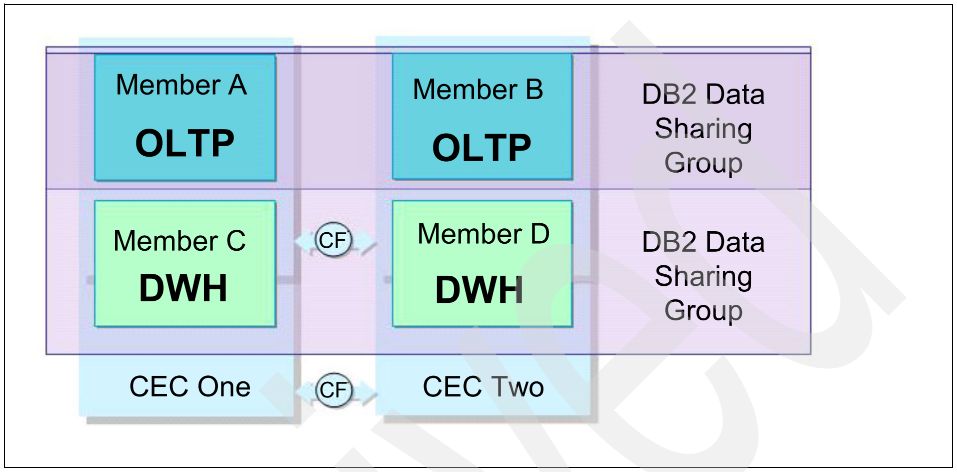


Figure 6-28 Centralized DWH on System z with DB2 Data Sharing

Keeping your most crucial data on the same platform simplifies compliance. The ability to monitor and log systems access and use can be much simplified on a central platform, compared with the complexity of documenting compliance across numerous distributed computing applications and systems.

Valid reasons for centralizing the DWH on System z include when:

- You are dealing with highly sensitive information.
- Most of the data (including DB2, VSAM, IMS, an so forth) feeding the warehouse resides on z/OS and is being continuously updated or added to.
- Access to data is required in almost real time.
- The organization has System z skills readily available. This is mostly the case in large enterprises already running DB2 on System z.
- Sufficient capacity is available on System z.
- Market trends in DWH solutions and vendors indicate that System z is a viable platform for DWH solutions.
- You want to reduce complexity due to the sheer volume of data involved and the associated issues if it must be replicated and moved around.

- Mergers and acquisitions or consolidations are taking place and, for example, System z in the new joint venture has dominance, or the merged companies wish to reduce the number of platforms and servers they currently maintain.
- The majority of the data is on System z and is constantly being added to, updated, changed.
- The data volume is too large and volatile to re-host.
- Business processes and business drivers require it, such as when customer infrastructure (intranet or Internet access, for example) is set up for System z with high affinity to warehouse data.
- The company wants or needs extremely high RAS in the DWH environment.

For more information about using DB2 on z/OS for DWH, refer to
`ftp://ftp.software.ibm.com/software/data/db2bi/data_warehousing_whitepaper.pdf`

6.7.2 Common location patterns for Business Intelligence (BI)

Pure hybrid BI/DWH solution

In this hybrid solution the DWH is on System z, but the BI solution runs on another distributed hardware platform. Because the data storage and processing remains on the mainframe, the benefits of the centralized DWH remain.

There are a large number of solutions for analysis and reporting that can be used for BI reporting on a Data Warehouse that is implemented with DB2 for z/OS. Examples of such solutions are Hyperion, Business Objects, SAS, and IBI; IBM also has several solutions, including Alphablox, DB2 QMF, DataQuant, and Cognos®.

Hybrid BI/DWH solution on System z

This solution has the same architecture as the pure hybrid solution, but the BI solution runs on Linux on System z.

The Integrated Facility for Linux (IFL) special engine allows Linux workloads to run on the System z platform without incurring any IBM software license charges. The use of an IFL could permit an organization to run any Linux-dependent components of the BI solution on the mainframe with all of the attendant security, reliability, and manageability hardware benefits of System z.

IBM DataQuant for WebSphere, IBM Alphablox, and Cognos BO are available for Linux on System z.

The main reasons for placing the BI application on System z are when you need:

- Low latency. You can make use of Hipersockets to access the data, which is many times stored on z/OS.
- Strong security and data privacy to address regulatory compliance.
- High reliability and availability of the hardware.

Homogeneous BI/DWH solution on z/OS

The only BI solutions that can be deployed on z/OS are IBM QMF and Dataquant for WebSphere. In this configuration IBM QMF and DataQuant make use of all Qualities of Service that WebSphere Application Server on z/OS provides.

The main reasons for placing the BI application on z/OS are when you need:

- Co-location with transactional data for low latency, reduced complexity, and lower cost.
- The strongest security and data privacy to address regulatory compliance.
- The highest reliability and availability.

In the following sections we discuss and compare some existing IBM solutions.

6.7.3 IBM QMF

DB2 QMF for WebSphere extends key QMF functionality to browser-based users across multiple platforms. For complete details regarding DB2 QMF for z/OS refer to:

`http://www.ibm.com/software/data/qmf/`

The features and capabilities of DB2 QMF for WebSphere include:

- New OLAP query builder, complete with graphical designer, hierarchical and dimension filtering, informational SQL view, and fully interactive results grid.
- New object repository, which simplifies the storage, access control, and distribution of QMF objects across the enterprise.
- It extends key QMF functionality to browser-based users across multiple platforms, including report authoring and drag-and-drop data analysis.
- The ability to develop and export query results and reports in a variety of formats including XML, HTML, and PDF.
- An improved table editor provides a means of directly inserting, deleting, and editing data in all supported database types.

6.7.4 IBM DataQuant

IBM DataQuant is a business analytics tool, built on Eclipse, which delivers a comprehensive query, reporting, and data visualization platform.

While remaining compatible with the IBM Query Management Facility (QMF), DataQuant introduces a variety of powerful business intelligence capabilities, including:

- Self-service reporting

  Empower end users with the ability to answer their own business questions while also decreasing the demands on the IT organization.
- Turnkey solution

  Powerful graphical reporting environment that allows rapid development and deployment of executive dashboards, information portals, and interactive data visualization solutions.
- Perform analytics

  Create graphical reports and interactive visual solutions on distributed or z/OS-based data warehouses, without moving data, with 100 built-in mathematical and analytical functions.
- Share information

  SOA infrastructure, enabling enterprises to share specific business intelligence content and data with partners over secure Internet or intranet connections.
- Distributed access

  Concurrently draw data from multiple relational data stores, such as multiple instances of DB2 for z/OS, DB2 for Linux, UNIX and Windows, and Informix®. DataQuant provides a comprehensive query, reporting and data visualization platform and introduces a wide variety of business intelligence capabilities, from executive dashboards and interactive visual applications to information-rich graphical reports and ad-hoc querying and data analysis.

DataQuant provides 2 components:

- DataQuant for Workstation: An Eclipse-based environment for the development of query, report, and dashboard solutions.
- DataQuant for WebSphere: A runtime environment capable of displaying DataQuant content using a “thin client” model.

DataQuant for WebSphere runs on any WebSphere Application Server supported OS, including z/OS and Linux on System z.

Figure 6-29 shows an overview of DataQuant.

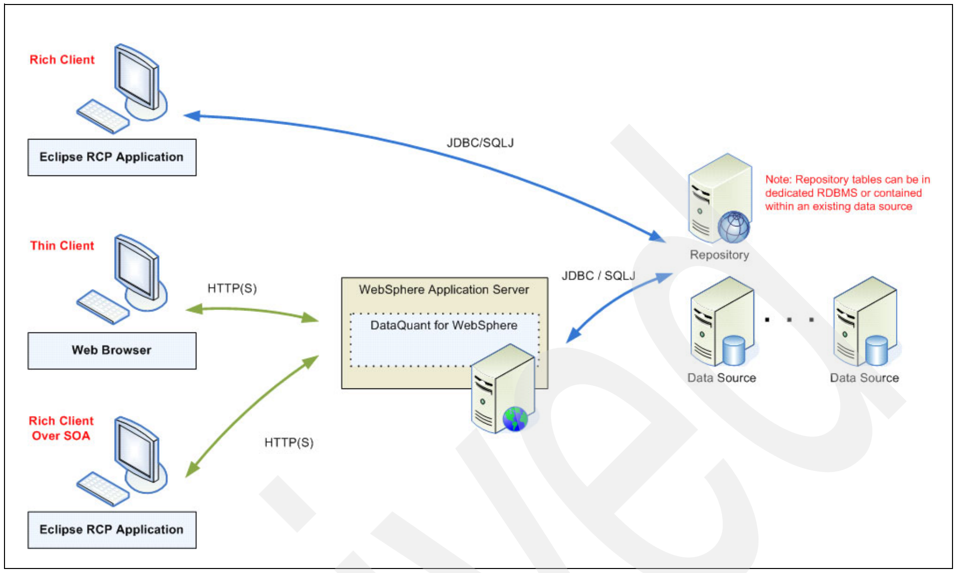


Figure 6-29 IBM DataQuant

When to consider DataQuant

The following are reasons to consider using DataQuant:

- When content authors want to quickly develop interactive dashboards and graphical page-based reports.
- Where there is a need for a turnkey solution that allows organizations to rapidly create and distribute Business Intelligence queries, reports, and interactive dashboards across the enterprise.
- Where there is a need for self-service reporting, empowering lines of business and end users with the ability to refine or design their own reports and dashboards, reducing the burden on IT.
- When resource governing, granular security, and object tracking are considered to be important factors in an enterprise-wide Business Intelligence solution.
- Where quick prototyping and rapid solution development is more important than complex analytical features.
- When there are already reporting capabilities implemented with QMF. DataQuant can re-use existing QMF queries, reports, and procedures — or interoperate with existing QMF infrastructure.

6.7.5 IBM Alphablox

IBM Alphablox is an application that executes in a Web application server environment, such as WebSphere Application Server.

It is a platform for the rapid assembly and broad deployment of integrated analytics embedded within applications. It has an open, extensible architecture based on the Java 2 platform, Enterprise Edition (J2EE) standards, and an industry standard for developing Web-based enterprise applications. Alphablox leverages AJAX technology, automatically handling many details of application behavior without the need for complex programming and providing an ultra-rich user experience with a zero footprint DHTML client.

Alphablox V9.5 is available for Linux on System z.

When to consider Alphablox

The following are reasons to consider using Alphablox:

- Where there is a need for customized analytical solutions that are tightly integrated and embedded within the existing BI infrastructure.
- Where “in-line” analytics are required, meaning where analytical application content is embedded in a business process.
- When there is a strong requirement for full control of the solution's appearance and behavior, from data gathering and analysis to the look and feel of the user interface.
- Where there is a need to include sophisticated analytical capabilities.
- As a development environment for customized Web applications instead of packaged ones.

6.7.6 Cognos 8 BI for Linux on System z

Cognos provides a complete, integrated, open standards-based platform for turning data into actionable insight for business intelligence and performance management, and leverages SOA to deliver greater agility.

These components share one common infrastructure for creating, managing, and deploying queries, reports, analyses, dashboards, scorecards, and alerts. IBM Cognos 8 Business Intelligence helps to meet an organization’s Business Intelligence objectives and the needs of every kind of user from simple report consumers to professional authors and developers.

Reporting with IBM Cognos 8 BI gives users the ability to author, share, and use reports that draw on data across all enterprise sources to inform better business decisions.

For more details, see the Cognos 8 BI Web site at:
`http://www.cognos.com/products/cognos8businessintelligence/index.html`

When to consider Cognos BI

The following are reasons to consider using Cognos 8 BI:

- When the previously mentioned solutions aren't able to solve the business request.
- Where you need to offering broader capabilities and enterprise reach than the previously mentioned solutions.

Frameworks and proven practices from Cognos accelerate the time to solution success.

6.8 Information Integration services solutions

In this section we discuss several Information Integration services solutions.

6.8.1 Patterns for Information Integration services

Figure 6-30 on page 323 shows a number of common information integration patterns, which we discuss in the following sections.

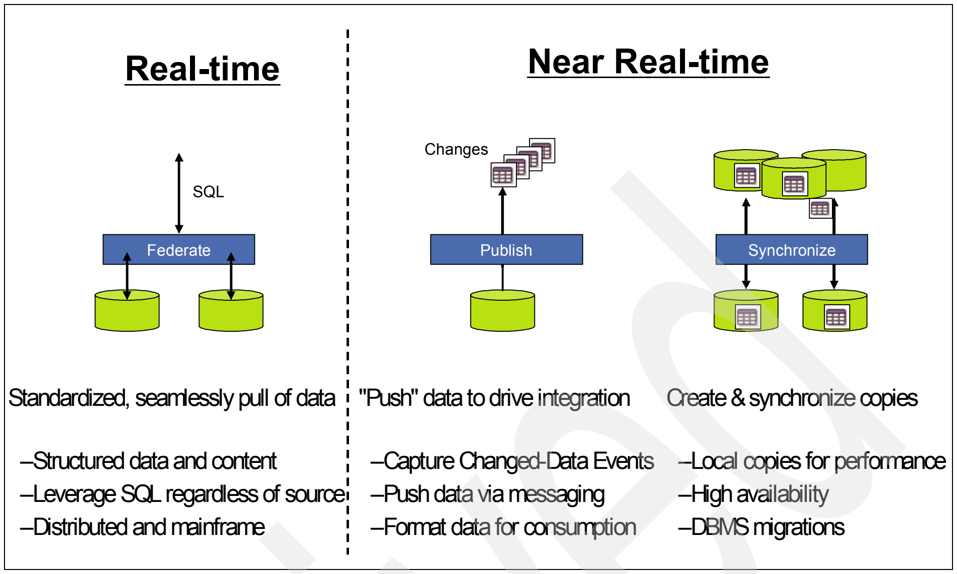

Real-time
Near Real-time
SQL
Federate
Changes
Publish
Synchronize
Standardized, seamlessly pull of data
–Structured data and content
–Leverage SQL regardless of source
–Distributed and mainframe
"Push" data to drive integration
–Capture Changed-Data Events
–Push data via messaging
–Format data for consumption
Create & synchronize copies
–Local copies for performance
–High availability
–DBMS migrations

Figure 6-30 Information Integration delivery patterns

Federation runtime pattern

Federation is a fundamental pattern that can be used in a number of ways, typically in support of another pattern. Its basis is a federated query that hides the fact that its target data might be local, remote, or fully distributed.

The federation pattern is a basic information integration application pattern that provides access to many diverse data sources and provides the appearance that these sources are a single logical data store.

The *data integration node* is at the core of the federation runtime pattern. It contains the knowledge and functionality that enables real-time access to and manipulation of data in disparate locations and formats across multiple data services nodes. The function of the data integration node is invoked by a consumer, who can be entirely separate, or integrated with the data integration functionality.

The data source is a generic data storage node that provides managed, persistent storage of any type of data and a means to directly access and manipulate that data. The data can be stored in files and accessed through file I/O routines or it can be stored in a database with more structured and managed access methods.

The flow is as follows:

1. A requesting consumer makes a query of data from the “federated” data source, for example, an HTTP/SOAP request.
2. The information service processes the request, and utilizing its metadata (which defines the data sources), passes on the requests to the appropriate data sources.
3. In many cases, the data integration/federation logic can be logically separate from the data connector logic. This data connector logic spreads out the overhead of making the query to multiple data sources, allowing the queries to run in parallel against each database. When performance is of major concern, multiple logical data connectors can exist to process queries against a single data source, the idea here being to prevent any single node in the process from becoming a bottleneck if too many requests run against one data source.
4. In all cases, the results that are returned from each individual data source must then be aggregated and normalized by the data integration layer so that these results appear to be from one “virtual” data source.
5. The results are then sent back to the requesting application, which has no idea that multiple data sources were involved.

In an SOA context, the federation server can act as a service provider or a service consumer, or both, which leverages SOA conforming interfaces. Note that this does not preclude the server from also providing support for the traditional, relational interfaces. When the data federation server exposes integrated information as a service provider, a service consumer can access the integrated information through a service interface such as WSDL and HTTP/SOAP or other agreed-to bindings. The data federation server can consume, in order to integrate, services provided by multiple information sources.

The thought behind using the data federation pattern in the SOA context is to leverage and reuse integrated information, that is, information integration services in an extensible manner for a variety of consumers. It is a commonly acknowledged best practice to design services so that they provide some combination of reuse, cross-enterprise interoperability, and business process enablement of information or functionality. Many if not most successful SOA projects focus first on the most important, most widely used business functions that are exposed as services. Due to the key role that those services play, they often span multiple back-end systems. Gathering information from multiple heterogeneous sources is therefore an important requirement and capability that SOA relies on. The service is not a query in the traditional data access context; rather, it is a request for a business entity (or entities) which might be fulfilled by the federation service through a series of queries and other services.

Publish runtime pattern

Data event publishing captures changes from one data source and publishes them for use in other data stores or by other applications.

It detects data changes in source databases, and copies those changes (if necessary, transformed), onto message queues where they can be consumed by other processing.

This pattern is particularly important where the source application should be relieved of the need to know anything about its targets. The primary IT driver for selecting this pattern is to allow loose coupling of clients and services with minimum modification to each.

Synchronization runtime pattern

Data replication covers the need to synchronize data between sources. Figure 6-31 illustrates different uses of replication for distribution to external systems, for consolidation on a System z platform, and for "cohabitation," a situation where updates can occur on both sides.

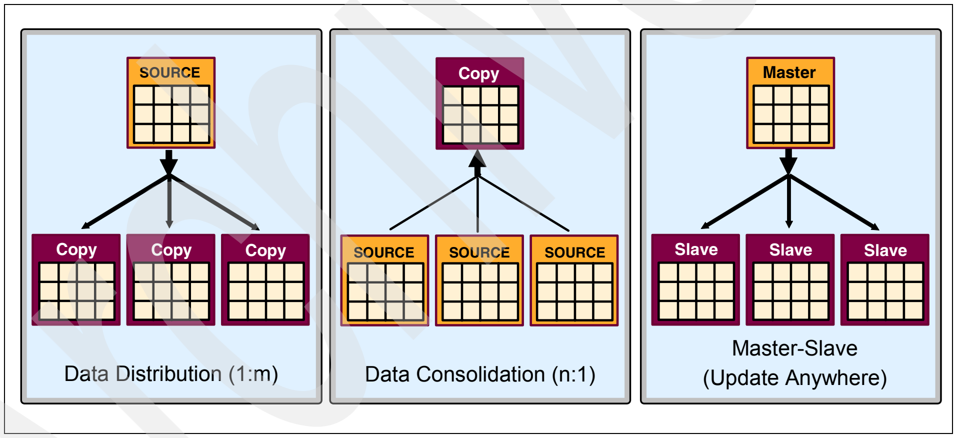


Figure 6-31 Synchronization runtime pattern

Synchronization has in common with Publish the activity of data (or information) being copied and, if necessary, transformed, from one place to another off line, in advance of when it is required by a user or application. It differs from Publish in that with Synchronization, data can flow in both directions. Publish is clearly the simpler of the two patterns. A significant issue in this case is conflict detection and resolution when updates occur independently in the different data stores.

The main purpose of this pattern is to create local copies for performance reasons, increased availability, or for use for DBMS migrations.

The synchronize pattern doesn't play an important role in the SOA story because the main reasons for the existence of this pattern are technical reasons and not business related. If you want to implement this pattern you can use WebSphere Classic Replication Server for z/OS or WebSphere Replication server for z/OS.

Data transformation runtime pattern

Data transformation converts source data to the different formats needed by the target systems.

As shown in Figure 6-32, the data transformation pattern is decomposed into its three primary constituents or steps:

- Gather
- Process
- Apply

Eventually this process can occur in a single step.

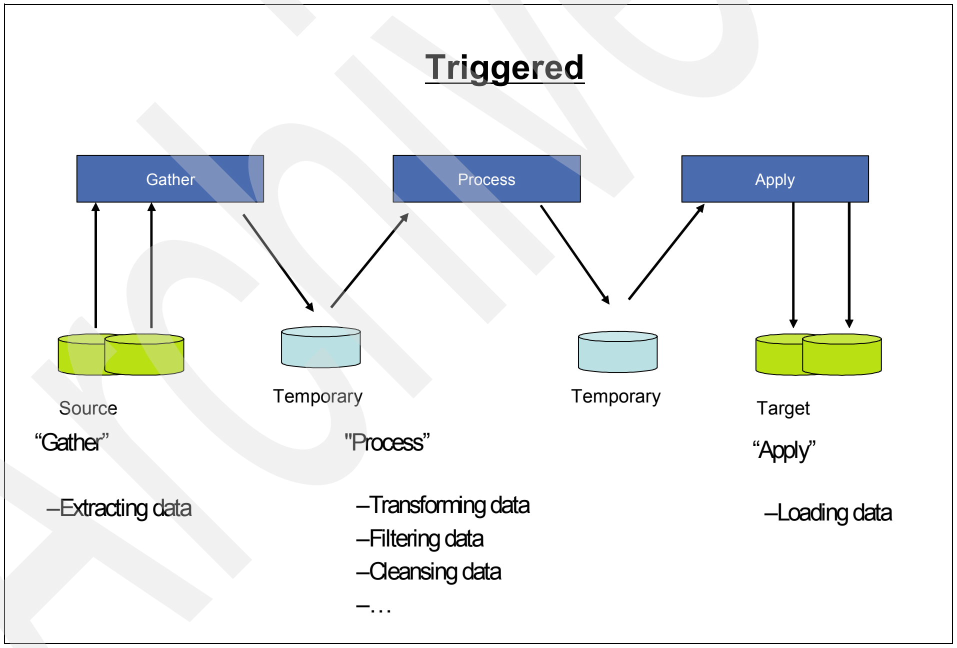


Figure 6-32   Data Transformation runtime pattern

The *Gather* step extracts the data from a data store that provides managed, persistent storage of any type of data and a means to directly access and manipulate that data. The data can be stored in files and accessed through file I/O routines or it can be stored in a database with more structured and managed access methods. The first step is designed and optimized for reading the data from the data store.

The *Process* step transforms the data, often in sophisticated ways, while it passes through. Eventually, other specialized Process steps can be added to handle the data under different circumstances, such as for efficient throughput of large batches of records that require extensive transformation, or for fast throughput of individual records in near real time.

Multiple data sources can be involved in the base runtime pattern process and reasonably sophisticated filtering, cleansing, and transformations can occur within the Process function.

Finally, in the *Apply* step, the data is loaded and data is written to data stores.

From an SOA point of view this pattern is used to re-purpose existing data for new business uses and service-enable cleansing processes for dynamic quality.

This might involve the use of *Extract-Transform-Load (ETL)* mechanisms for processing and enriching data in bulk, batch processing activities involved in bulk transaction processing, and migrating data from master-data-of-record databases to information warehouses that can be used to perform postprocessing and business intelligence and content management functions, which in turn are made available to the business application as services.

6.8.2 IBM Information Server for System z

IBM has created a platform for hosting the solutions that implement information integration services. This platform enables information integration functions to participate in an SOA world more efficiently and helps by delivering these services in an “always on” fashion. Delivering these services in this fashion gives the ability to integrate information integrity functions as services in complex business processes. Figure 6-33 on page 328 shows the vision.

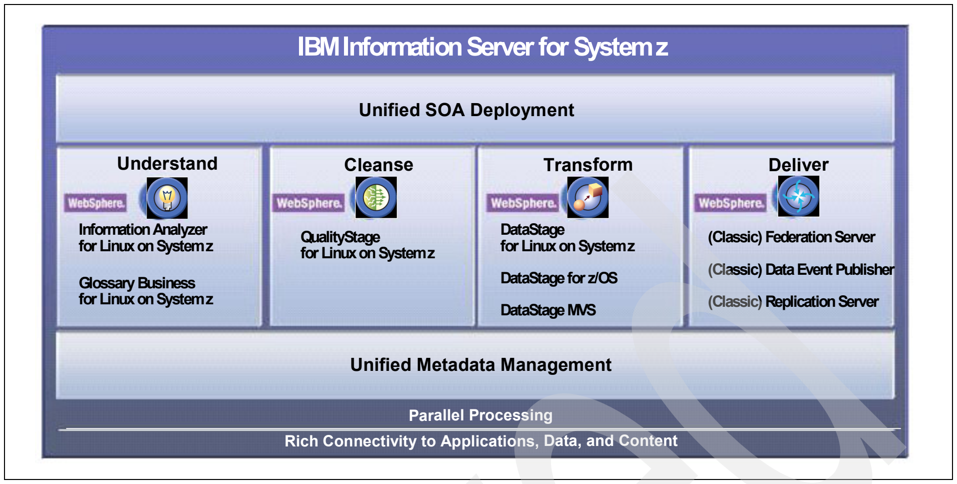

IBM Information Server for System z
Unified SOA Deployment
Understand
WebSphere.
Information Analyzer for Linux on System z
Glossary Business for Linux on System z
Cleanse
WebSphere.
QualityStage for Linux on System z
Transform
WebSphere.
DataStage for Linux on System z
DataStage for z/OS
DataStage MVS
Deliver
WebSphere.
(Classic) Federation Server
(Classic) Data Event Publisher
(Classic) Replication Server
Unified Metadata Management
Parallel Processing
Rich Connectivity to Applications, Data, and Content

Figure 6-33 IBM Information Server for System z

Figure 6-34 shows the solutions available.

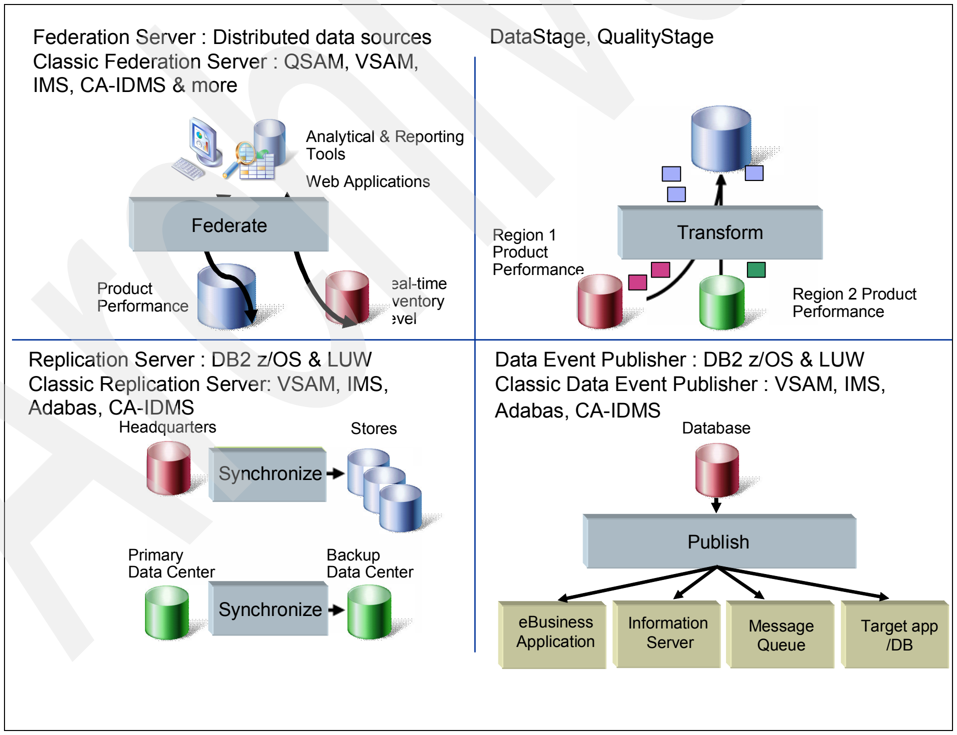

Federation Server : Distributed data sources
Classic Federation Server : QSAM, VSAM, IMS, CA-IDMS & more
Analytical & Reporting Tools
Web Applications
Federate
Product Performance
eal-time ventory evel
DataStage, QualityStage
Transform
Region 1 Product Performance
Region 2 Product Performance
Replication Server : DB2 z/OS & LUW
Classic Replication Server: VSAM, IMS, Adabas, CA-IDMS
Headquarters
Stores
Synchronize
Primary Data Center
Backup Data Center
Synchronize
Data Event Publisher : DB2 z/OS & LUW
Classic Data Event Publisher : VSAM, IMS, Adabas, CA-IDMS
Database
Publish
eBusiness Application
Information Server
Message Queue
Target app /DB

Figure 6-34 Solutions for information integration services overview

The IBM *WebSphere Information Services Director* (Figure 6-35 on page 330) provides:

- Access to a range of information sources with various bindings such as SOAP/HTTP for Web services, or Enterprise Java Beans (EJBs) for high-speed direct Java integration.
- A resilient infrastructure that supports both fault tolerance and load balancing, thereby satisfying the high availability requirements.
- Access to a range of integration functionality, including cleanse, transformation, federation, replication, and event publishing.
- The most flexibility in how these functions are used, including support for service-oriented architectures, event-driven processing, scheduled batch processing, and even standard APIs like SQL and Java.
- Secure and resilient data integration services to J2EE applications and databases.
- A services catalog maintaining all available services built on the MetaData Server infrastructure. The services catalog has an out-of-the-box integration with WebSphere Service Registry and Repository (WSRR) to seamlessly publish all services into the enterprise-wide service repository.

The IBM WebSphere Information Services Director can be hosted in a WebSphere Application Server environment on Linux on System z.

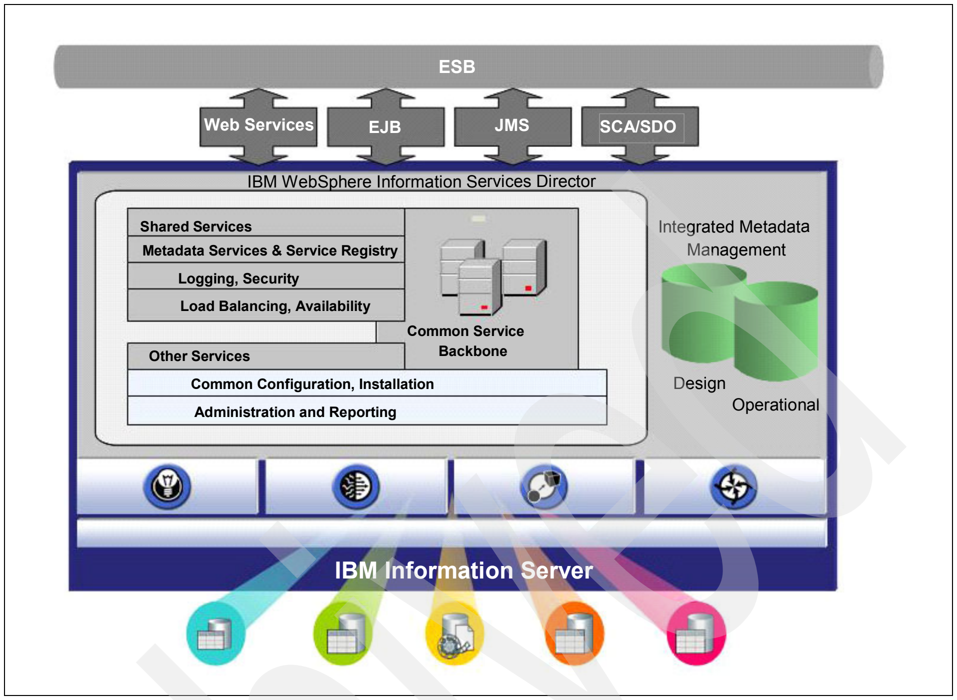

ESB
Web Services
EJB
JMS
SCA/SDO
IBM WebSphere Information Services Director
Shared Services
Metadata Services & Service Registry
Logging, Security
Load Balancing, Availability
Common Service Backbone
Other Services
Common Configuration, Installation
Administration and Reporting
Integrated Metadata Management
Design
Operational
IBM Information Server

Figure 6-35 Common programming model WebSphere Information Services Director

6.8.3 Common solutions for federation pattern

There are several solutions available to implement the federation pattern on System z, including building your own solution. Figure 6-36 on page 331 shows an overview of the most important solutions on System z. The solutions we discuss further are:

- IBM InfoSphere Classic Federation Server for z/OS
- WebSphere Federation Server, available for Linux on System z
- WebSphere Information Integrator Content Edition, available for Linux on System z

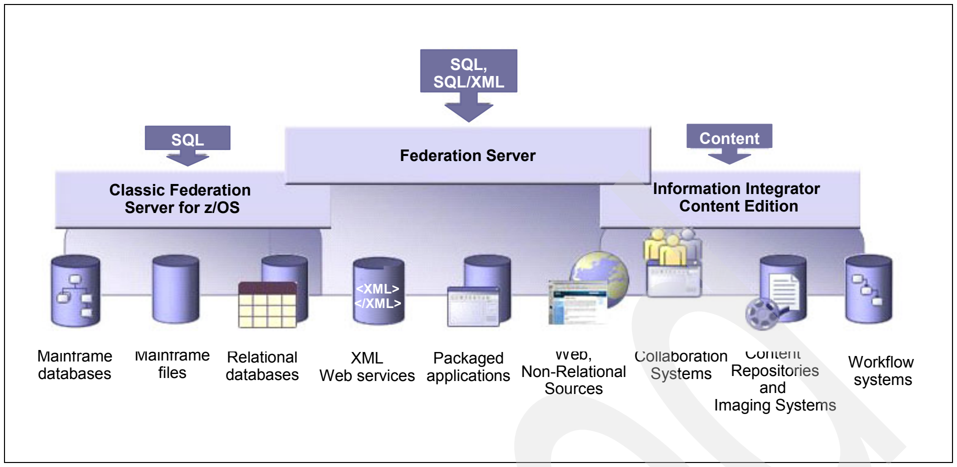


Figure 6-36 Solutions for Federation pattern overview

IBM *InfoSphere Classic Federation Server for z/OS* can help you to interact with data services that are on System z. Having the interaction service on z/OS has the benefit that it is co-located with the data for low latency, reduced complexity, the strongest security and data privacy to address regulatory compliance, and high reliability and availability. More detailed information can be found at:

```
http://www-01.ibm.com/software/data/infosphere/classic-federation-se
rver-z/
```

If you want to federate DB2 access on z/OS with diverse and distributed data on other platforms, *WebSphere Federation Server* helps to get there. WebSphere Federation Server can be implemented on Linux on System z to benefit from the RAS hardware features of System z. More detailed information can be found at:

```
http://www-01.ibm.com/software/data/integration/federation_server/
```

WebSphere Information Integrator Content Edition can give you federated access to disparate content management systems. This solution can be eventually implemented in a WebSphere Application Server Network Deployment configuration on Linux on System z. More detailed information can be found at:

```
http://www-01.ibm.com/software/data/integration/db2ii/editions_conte
nt.html
```

InfoSphere Classic Federation Server for z/OS

InfoSphere Classic Federation Server for z/OS provides direct, real-time SQL access to mainframe databases and files without mainframe programming. It maps logical relational table structures to existing physical mainframe databases and files. This product is the successor of IBM *WebSphere Information Integrator Classic Federation*.

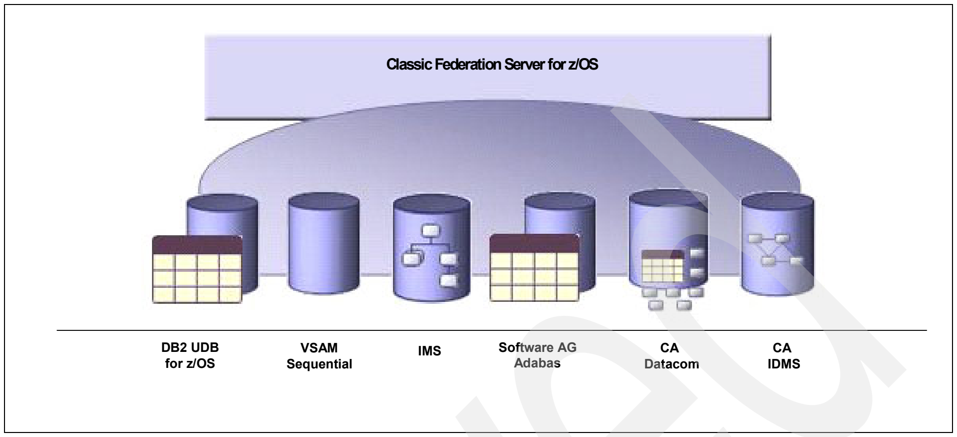


Figure 6-37 InfoSphere Classic Federation for z/OS

InfoSphere Classic Federation Server for z/OS can only be accessed via SQL and not directly via Web Services or SOAP/HTTP.

InfoSphere Classic Federation Server for z/OS dynamically generates native data access commands that are optimized for each database and file type. Results are automatically translated and reformatted into relational rows and columns, providing seamless integration of all mainframe data assets without proprietary programming.

The business benefits of InfoSphere Classic Federation Server for z/OS V9.5 include:

- Extending the value of existing mainframe investments. Rather than ripping out and replacing mainframe assets, clients can extend these investments to deliver new business applications or integrate business operations.
- Reducing time-to-value for integration projects that require mainframe data by eliminating proprietary coding and multi-platform development and automating complex multi-source integration.
- Delivering up-to-the second information rather than stale copied data.
- Working with the most widely used mainframe security systems, known management and monitoring tools, as well as commonly used accounting practices.
- Leveraging an SAF exit for RACROUTE calls. While this is syntactically a RACF command protocol, both ACF2 and TopSecret support this protocol for compatibility purposes.

- DB2-like security can also be added via metadata. The logical tables and views can have privileges (read only, updateable, and so forth) attached and assigned to specific users.
- Interoperating with existing tools while also providing some tools of its own. Its operational environment can be monitored using typical system management tools that every organization already has implemented. And, “governors” can be set in the product configuration. These governors will trigger data access errors when a user exceeds their authorized CPU time utilization or their authorized data access limits. This protects the system from the “query from hell.” And, these governors can be set at different levels: system, server, and user level.
- Possibility to charge back the cost of the mainframe resources directly to the business units. There are two facilities provided. First, a site-specific SMF exit routine can be specified. Second, InfoSphere Classic Federation Server can optionally create SMF type 99 records. These records can then be processed by an installation-specific accounting routine to determine chargeback or for other utilization monitoring purposes.
- Tracing and logging can be set at multiple levels. Initiation of tracing and logging can also be triggered locally or remotely. These features can be triggered (that is, real time driven) or utilized for postmortem processing.

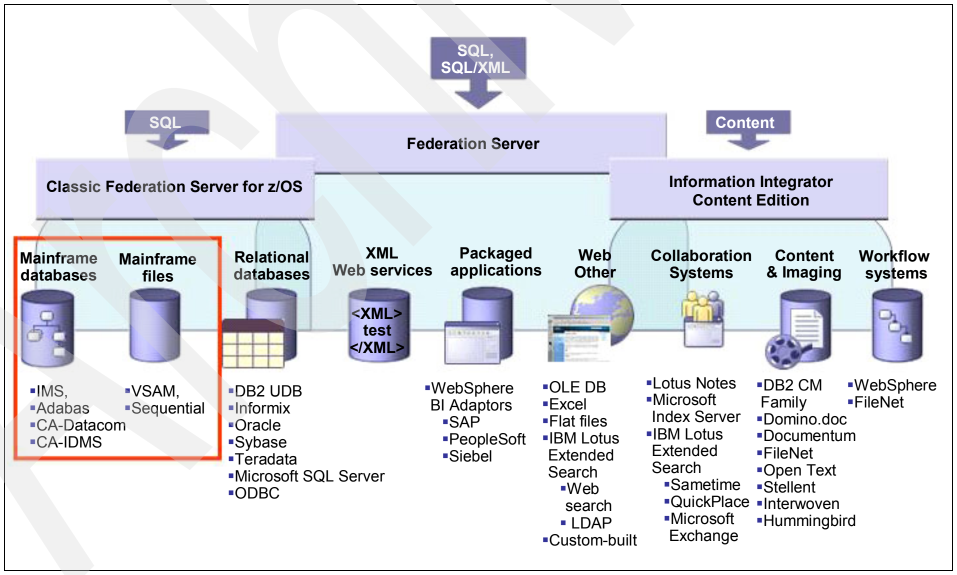

SQL, SQL/XML
SQL
Federation Server
Content
Classic Federation Server for z/OS
Information Integrator Content Edition
Mainframe databases
Mainframe files
Relational databases
XML Web services
Packaged applications
Web Other
Collaboration Systems
Content & Imaging
Workflow systems
<XML> test </XML>
▪IMS, ▪Adabas ▪CA-Datacom ▪CA-IDMS
▪VSAM, ▪Sequential
▪DB2 UDB ▪Informix ▪Oracle ▪Sybase ▪Teradata ▪Microsoft SQL Server ▪ODBC
▪WebSphere BI Adaptors ▪SAP ▪PeopleSoft ▪Siebel
▪OLE DB ▪Excel ▪Flat files ▪IBM Lotus Extended Search ▪Web search ▪LDAP ▪Custom-built
▪Lotus Notes ▪Microsoft Index Server ▪IBM Lotus Extended Search ▪Sametime ▪QuickPlace ▪Microsoft Exchange
▪DB2 CM Family ▪Domino.doc ▪Documentum ▪FileNet ▪Open Text ▪Stellent ▪Interwoven ▪Hummingbird
▪WebSphere ▪FileNet

Figure 6-38 InfoSphere Classic Federation Server for z/OS

InfoSphere Classic Federation Server for z/OS can also extend IBM WebSphere Federation Server V9.1 access to non-relational mainframe data sources.

WebSphere Federation Server lets applications access diverse and distributed data, including multivendor mainframe and distributed, structured and unstructured, public and private, as though it were a single database.

InfoSphere Classic Federation Server for z/OS v9.1 supports the following host databases on z/OS:

- Software AG Adabas
- CA-Datacom
- Advantage CA-IDMS/DB for z/OS
- IBM DB2 Universal Database™ for z/OS
- IMS
- VSAM for z/OS
- Sequential files for z/OS

The architecture of InfoSphere Classic Federation Server for z/OS consists of the following major components:

- Data servers

  Data servers perform all data access. The architecture of the data server is service-based. The data server consists of several components, or services. A major service embedded in the data server is the query processor that acts as the relational engine for Classic federation.

- Data connectors

  The query processor dynamically loads one or more data connectors to access the target database or file system that is referenced in a SQL request.

- Classic Data Architect

  To process SQL data access requests, data definitions must be mapped to logical tables. The Classic Data Architect is the administrative tool that you use to perform this mapping.

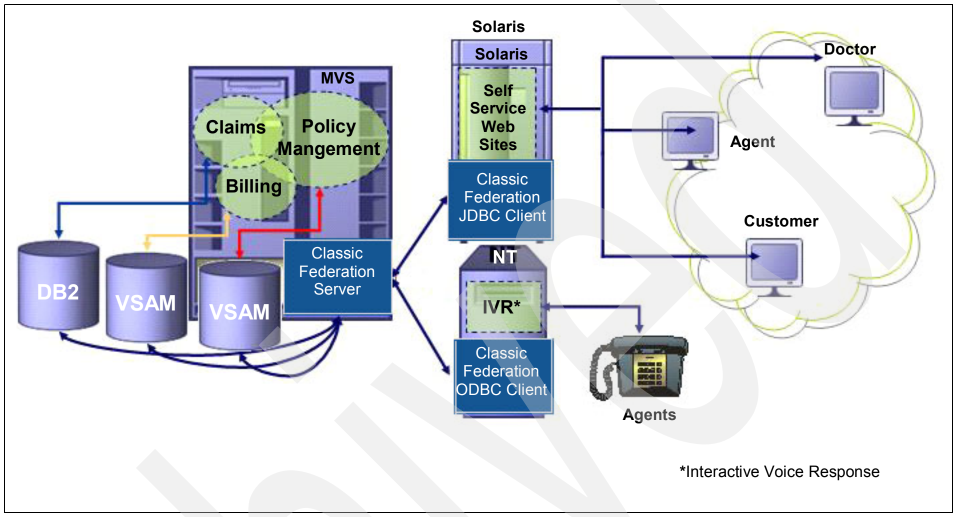
- Metadata catalog

  The information that you generate from the Classic Data Architect is stored in metadata catalogs. A metadata catalog is a set of relational tables that contain information about how to convert data from non-relational to relational formats. The data server accesses the information stored in these catalogs.

- Clients

  InfoSphere Classic Federation Server for z/OS provides the ODBC, JDBC, and CLI clients. The clients enable client applications or tools to submit SQL queries to the data server.

Figure 6-39 shows a sample implementation of InfoSphere Classic Federation Server's predecessor for a large insurance carrier. The WebSphere II Classic Federation solution directly connects the Interactive Voice Response[1] system and the Self Service Web site to the mainframe-based operational databases and files.



Solaris
Solaris
Self Service Web Sites
Classic Federation JDBC Client
MVS
Claims
Policy Mangement
Billing
DB2
VSAM
VSAM
Classic Federation Server
NT
IVR*
Classic Federation ODBC Client
Doctor
Agent
Customer
Agents
*Interactive Voice Response

Figure 6-39 Sample implementation of Classic Federation Server

WebSphere Federation Server

IBM WebSphere Federation Server creates a consolidated view of your data to support key business processes and decisions.

WebSphere Federation Server enables applications to access and integrate diverse data and content sources as though they were a single resource, regardless of where the information resides, while retaining the autonomy and integrity of the data and content sources.

WebSphere Federation Server offers integrated views of a wide range of heterogeneous data and content sources. Applications use a standard SQL interface or standard APIs to query data and content sources.

WebSphere Federation Server fits neatly and transparently behind common analytical and reporting tools, development environments, portals, and other standard IT infrastructure components. For instance, business applications

[1] IVRs are those wonderful telephony systems that have you press or say "1" to get help with billing, press or say "2" to get information, and so forth.

access integrated views of existing and new information through an abstraction layer that insulates them from changes in the source material they are viewing.

WebSphere Federation Server provides virtualized access to enterprise information as though from a single source, while maintaining source integrity. Version 9.1 enhances previously available federation offerings with such features as federated two-phase commit, support for remote stored procedures, and expanded platform support. Two-phase commit adds the ability to update multiple data sources simultaneously within a distributed system. Remote stored procedures allow users to leverage existing investments in procedures for disparate data sources.

The features of WebSphere Federation Server include the capability to:

- Correlate data from local tables and remote data sources as though all the data is stored locally in the federated database
- Update data in relational data sources as though the data is stored in the federated database
- Move data to and from relational data sources
- Use data source processing strengths by sending requests to the data sources for processing
- Compensate for SQL limitations at the data source by processing parts of a distributed request at the federated server
- Access data anywhere in your enterprise, regardless of what format it is in or what vendor you use, without creating new databases and without disruptive changes to existing ones, using standard SQL and any tool that supports JDBC or ODBC
- Use visual tools for federated data discovery and data modeling
- Provide industry-leading query optimization with single sign-on, unified views, and function compensation
- Provide federated two-phase commit for updating multiple data sources simultaneously within a distributed system, while maintaining data integrity across distributed sources

The federated server

In a federated system, the server that receives query requests and distributes those queries to remote data sources is referred to as the *federated server*. A federated server embeds an instance of DB2 to perform query optimization and to store statistics about remote data sources.

Application processes connect and submit requests to the database within the federated server. A federated server is configured to receive requests that might be intended for data sources. The federated server distributes these requests to the data sources.

A federated server uses the native client of the data source to access the data source. For example, a federated server uses the Sybase Open Client to access Sybase data sources and a Microsoft SQL Server® ODBC Driver to access Microsoft SQL Server data sources.

The federated database

To users and client applications, data sources appear as a single relational database. Users and applications interface with the federated database that is managed by the federated server.

The federated database system catalog contains entries that identify data sources and their characteristics. The federated server consults the information that is stored in the federated database system catalog and the data source connector to determine the best plan for processing SQL statements.

The federated system processes SQL statements as though the data from the data sources was ordinary relational tables or views within the federated database.

The federated system can correlate relational data with data in non-relational formats. This is true even when the data sources use different SQL dialects, or do not support SQL at all.

The characteristics of the federated database take precedence when they differ from the characteristics of the data sources. Query results conform to DB2 semantics, even if data from other non-DB2 data sources is used to compute the query result.

WebSphere Information Integrator Content Edition

WebSphere Information Integrator Content Edition delivers federated access to the vast majority of enterprise information that lives in disparate content management systems. It makes IBM and non-IBM sources look and act as a single unified repository, and provides a platform for deploying applications and workflows spanning multiple content sources.

IBM WebSphere Information Integrator Content Edition enables organizations to quickly unify these isolated content silos to improve access, reuse, and control of distributed content assets, and to dramatically reduce the cost and complexity associated with creating content-centric applications.

WebSphere Information Integrator Content Edition is based on Java technology and designed to run in a J2EE environment, such as IBM WebSphere Application Server.

It is based on a service-oriented architecture and provides access to disparate content repositories and workflow systems. WebSphere Information Integrator Content Edition is fully J2EE-compliant and Web services compatible.

Among the pre-built connectors for WebSphere Information Integrator Content Edition are:

- IBM Content Manager
- IBM Content Manager OnDemand
- WebSphere Portal Document Manager
- IBM WebSphere MQ Workflow
- IBM Lotus Notes
- IBM Lotus Domino Document Manager
- FileNet® Content Services
- FileNet Image Services
- FileNet Image Services Resource Adapter
- FileNet Content Manager
- Documentum Content Server
- Microsoft Index Server/NTFS
- Open Text Livelink
- Hummingbird Enterprise DM
- DB2 Universal Database
- Oracle

More information can be found at:

`http://www-01.ibm.com/software/data/integration/db2ii/editions_content.html`

6.8.4 Common solutions for Publish pattern

All these solutions need at least a part of the solution implemented on the same platform where the data store is located to detect the data changes. In almost all cases they are implemented on the same platform where the data store is located to avoid all the communication and data transfer overhead to publish the data before it can be consumed by other SOA services.

Figure 6-40 shows two solutions in this space:

- IBM InfoSphere Classic Data Event Publisher for z/OS
- IBM WebSphere Data Event Publisher, also available for z/OS as WebSphere Data Event Publisher for z/OS. This solution is discussed in “WebSphere Data Event Publisher for z/OS” on page 340.

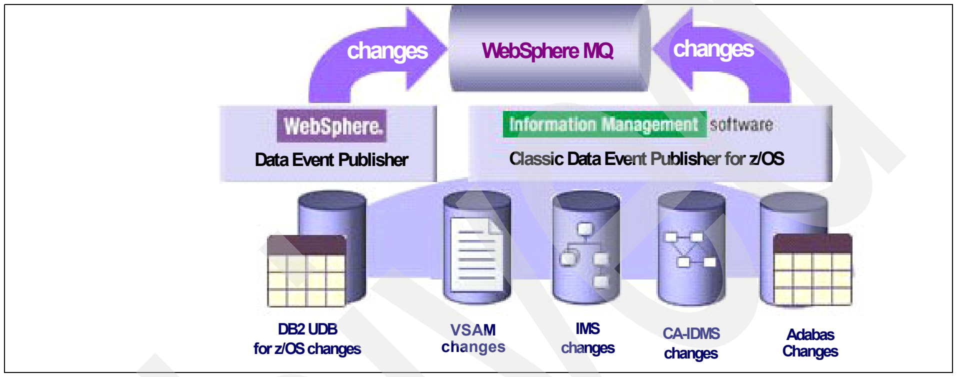


Figure 6-40 WebSphere Data Event Publisher solutions for z/OS

IBM InfoSphere Classic Data Event Publisher for z/OS

IBM InfoSphere Classic Data Event Publisher for z/OS provides very low latency response to events with high reliability, and allows mainframe data to trigger downstream processing. It captures data changes in real time, publishes “data events” to drive integration and incremental updating, and it enables integration using data events rather than custom application development.

The main benefits are:

- Remove data event capture processing from the operational application, thereby reducing overhead within the transaction path
- Low latency for scheduled data capture
- Multiple publication formats including XML for ease of consumption
- Recoverable
- Assured delivery
- Make the data event capture independent of the structure or processing flow of the underlying applications by a loosely coupled approach

The following data sources are supported:

- IMS
- Computer Associates CA-IDMS
- Software AG Adabas
- VSAM

WebSphere Data Event Publisher for z/OS

IBM WebSphere Data Event Publisher for z/OS has the same functionality as the InfoSphere Classic Data Event Publisher for z/OS for DB2. It eliminates the hand coding typically required to detect DB2 data changes made by operational applications and links changed data events in DB2 for z/OS databases with your business application integration solutions.

It captures and publishes DB2 changed-data events such as WebSphere MQ messages that can be used by other applications and tools to drive subsequent processing.

You can, for example, publish data changes that can be consumed by other services for populating a data warehouse, data mart, or operational data store with the changes, or updating a CRM application.

6.8.5 Common solutions for Data Transformation pattern

The mainframe has a long tradition in the execution of transformation jobs in the traditional MVS environment.

In this pattern, data traffic plays an important role because this pattern is also used for batch processing activities involving bulk transaction processing. In this pattern, in most circumstances the response time plays a less important role because most of these services run asynchronously.

If your data store is located on z/OS the best solution is to have your Data Transformation services also on System z. IBM Information Server is the IBM platform for the implementation of Data Transformation services.

DataStage® and QualityStage for Linux on System z can provide a solution for the Data Transformation services implementation of the patterns explained in 6.3, “Common patterns” on page 278.

The Data Transformation services implementation of the pattern described in 6.3.3, “Information services on z/OS” on page 281 can be executed with DataStage for z/OS.

QualityStage for Linux on System z is a solution if the Processing step consists of cleansing.

DataStage for Linux on System z

The processes that manipulate the data (DataStage jobs) are hosted on Linux on System z. For most applications Linux on System z is an acceptable platform for hosting these processes because it takes advantage of the RAS characteristics of the System z hardware at a lower price compared with hosting them on z/OS. For most data transformation processes it is homogeneous enough (no physical network) and has acceptable connectivity features (Hipersockets) to deal with the throughput.

Figure 6-41 shows an information integration infrastructure with DataStage processes running on Linux on System z.

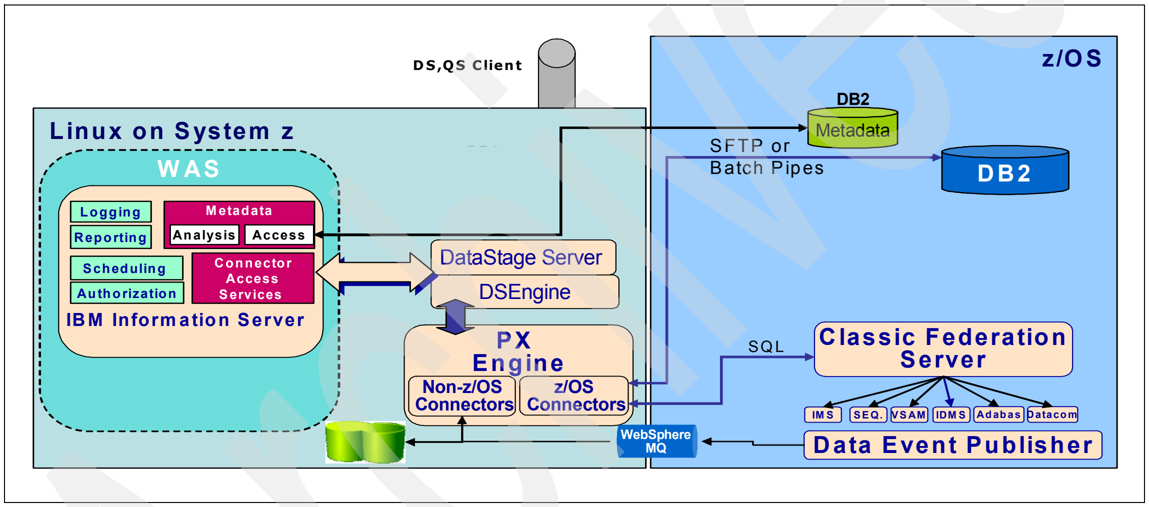


Figure 6-41 Datastage on Linux on System z

DataStage for z/OS

In this configuration the DataStage/QualtyStage processes are running in the UNIX System Services under z/OS.

Figure 6-42 on page 342 shows an information integration infrastructure with DataStage processes running on z/OS.

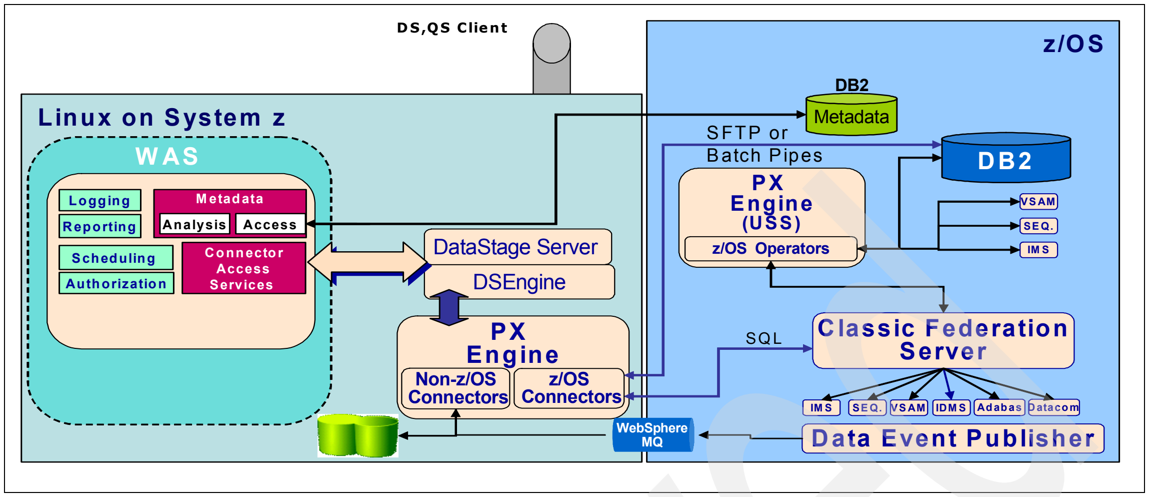


Figure 6-42 DataStage for z/OS and DataStage for Linux on System z

6.9 Master Data Management services solutions

IBM has recently released Master Data Management capabilities on System z in the form of InfoSphere Master Data Management Server for Linux on System z, which allows businesses to centrally manage customer, product, and account data for use across an enterprise.

New applications will be based on the new information repositories that will result from an MDM implementation project. Existing applications cannot stop using at once the specific data and formats they have been tied to since their design. Interim solutions are needed to make those applications run while data creation and updates are delegated to the new MDM system.

The MDM process helps analyze where the master data should reside. Thanks to various tools, this data is copied or propagated to existing data stores if needed.

The System z platform is uniquely positioned to take up the MDM challenge because it already hosts much reference data.

WebSphere Customer Center and IBM InfoSphere Master Data Management Server can leverage the z/OS platform as database platform.

7

SOA Service Management

The challenge today is to increase customer satisfaction, contain costs, maintain a secure environment and keep the enterprise in compliance in an IT environment where IT infrastructures are getting more complex, the rate of change is increasing, costs are raising, and service quality is difficult to maintain. Enterprises today require an IT Service Management strategy to define how to optimize service delivery while keeping quality, costs, security, and compliance in balance.

In this chapter, we discuss the definitions, scope and functions of IT Service Management with respect to SOA. We also describe several IT Service Management disciplines. In addition, we discuss some IT infrastructure solutions for each Service Management discipline. These solutions are designed to help manage the IT infrastructure and deliver IT services more effectively and efficiently.

7.1 Definitions, scope, and functions

In this section, we define and describe the IT Service Management scope and we describe several key functions of IT Service Management with respect to service-oriented architecture.

7.1.1 Definitions

Managing services and managing the overall SOA implementation are different, although overlapping problem areas. Loosely coupled services, after all, are simply interfaces; effective management of such interfaces necessarily requires management of the underlying infrastructure. This reality highlights one essential connection between IT Service Management and SOA Management: maintaining the loose coupling at the service interface requires ITSM, because loosely coupled services depend upon IT services. While loose coupling provides flexibility, governance helps to ensure the reusability of services, which is also a challenge for traditional systems management. Higher flexibility leads to higher rates of change, while loose coupling suggests more things to monitor and more relationships to track.

SOA management goes well beyond management of services. Specifically, SOA management also requires an increased ability to manage both business and IT processes, including business process monitoring change and release management, configuration management, availability management, security management, and role management.

IT Service Management represents the set of management tools used to monitor your service flows, the health of the underlying system, the utilization of resources, the identification of outages and bottlenecks, the attainment of service goals, the enforcement of administrative policies, and recovery from failures. IT Service Management is a supporting element of the logical architecture of SOA.

7.1.2 Scope

IT Service Management manages the elements that are deployed in the services architecture as discrete resources regardless of their implementation. While Service Management addresses many of the traditional problems of integrating disparate business processes and applications, deploying service-oriented applications introduces new complexities that must be managed. IT Service

Management is one of the building blocks in the SOA Reference Architecture (see Figure 7-1 on page 346). IT Service Management encompasses the following key functions:

- Discover all key elements of the services architecture, including discovering new service interfaces and identifying the relationships of those services to other services, the IT infrastructure, and business processes.
- Monitor the services layer against Service Level Agreements (SLAs).
- Collect performance and availability data to help define an SLA and provide problem identification.
- Respect and enforce management policies for the architectural elements of services management.
- Control the services management through configuration and operations.
- Provide notifications to the management application when states change or problems occur.
- Provide services security.
- Manage the life cycle of services in production, facilitating the graceful introduction and deprecation of services, including versioning capabilities.
- Provide root-cause analysis and problem resolution based on errors at the services layer.
- Transform and route messages based on message content and context, as well as policy (usually in support of SLA or version management).
- Track the business impact of services on the business processes that Web Services support.
- Manage the IT infrastructure that supports the services, associating problems in the infrastructure with the manifestation of problems at the service layer.

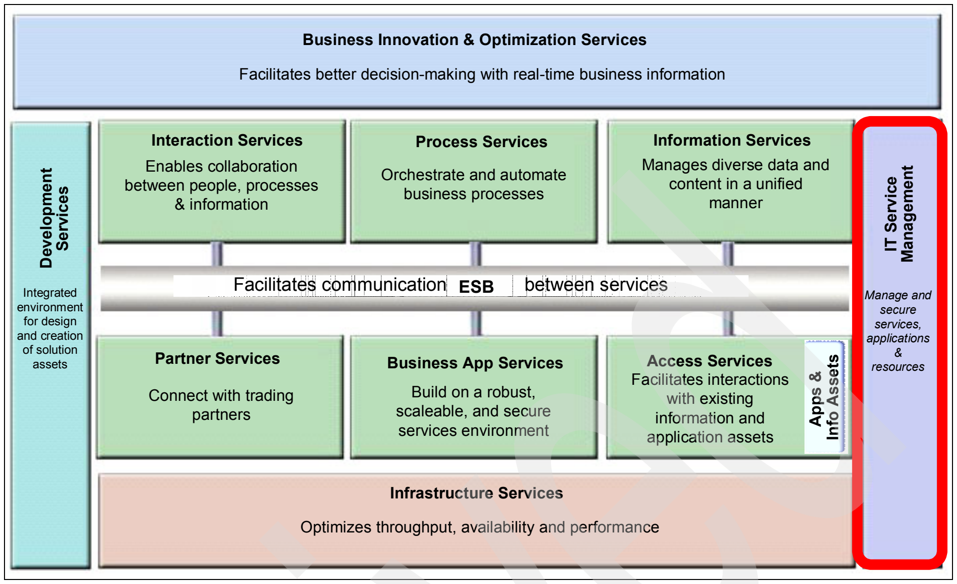

Business Innovation & Optimization Services
Facilitates better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enables collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manages diverse data and content in a unified manner
Facilitates communication ESB between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitates interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimizes throughput, availability and performance
IT Service Management
Manage and secure services, applications & resources

Figure 7-1 IT Services Management in the IBM SOA Reference Architecture

7.1.3 Functions

IT Service Management consists of the following management functions:

- Availability and performance management
- Security management
- Service assurance management
- Capacity management
- Service continuity management
- Asset and financial management
- Storage management

Availability and performance management

The purpose of availability and performance management is to match the availability and performance of the IT services against the current and future identified needs of the business or to exceed them. Availability and performance management enhances the availability of services by planning long-term service availability, measuring and monitoring service performance, and formulating service availability design criteria that meet requirements. These topics, except for design criteria, are discussed in detail in 7.2, “Availability and performance management” on page 348.

Security management

The open, diverse, and heterogeneous nature of an SOA environment presents unique challenges in terms of security, ease of use, and application management. To fully address these issues, a solution is required that spans all aspects of SOA. In 7.3, “Security management” on page 362, we discuss identity management, access management, federation management, and security compliance.

Service assurance management

For an IT organization, providing quality services is not enough. The service must consistently be of the same high quality both in actual delivery and in the eyes of the users of the services. *Service Level Management (SLM)* supports IT organizations to improve the quality of the services provided and the quality of the services as it is perceived by the users of IT services. The purpose of the Service Level Management process is to ensure that the service delivered to customers matches or exceeds the agreed, committed service quality characteristics. In 7.4, “Service Assurance management” on page 424, we discuss performance tracking and SLA management.

Capacity management

The purpose of capacity management is to match the capacity of the IT services and infrastructure to the current and future identified needs of the business. Capacity management focuses on the complete spectrum from design and planning of service capacities through the operational aspects of service capacity. For the purpose of this discussion we cover only the operational aspects of service capacity. This topic is presented in 7.5, “Capacity management” on page 430.

Service continuity management

The purpose of service continuity management is to focus on the IT services that will support business requirements in the event of a disruption to the business, based on the committed recovery schedule. This topic is discussed in 7.6, “Service continuity management” on page 434.

Asset and financial management

The purpose of asset and financial management is to ensure that financial controls and procedures are in place to effectively predict and control IT budgets, enable business decisions, and ensure that legal, corporate, and regulatory compliance is maintained. The outputs from the financial management process also enable benchmarking and business case analysis to support organizational decision making. In 7.7, “Asset and financial management” on page 443, we discuss the accounting management solution for SOA.

Storage management

The purpose of the storage management function is to get an enterprise view of storage capacity, utilization, and performance to optimize use and availability. It also enables better control of storage management costs by more effectively leveraging capacity and tiering of information storage resources. Storage management also encompasses backup and protection of critical applications and files to enable quick and automatic data recovery, establishing data protection policies that align with data availability service levels of the applications, and providing data recovery analysis in a comprehensive disaster recovery plan. Storing backup copies in a hierarchy of lower-cost storage, and the ability to easily manage backup and recovery across disk and tape from a single point of control are both critical to the overall SOA infrastructure.

7.2 Availability and performance management

In today's environment, businesses require IT to provide services for operation in a timely way. Any service interruption causes business loss and impacts customer satisfaction. Availability normally is the first priority consideration for IT services operation. Availability management understands the IT service requirements of the business. It plans, measures, monitors, and strives continuously to improve the availability of the IT infrastructure to ensure the agreed requirements are consistently met. Figure 7-2 on page 349 shows availability and performance monitoring in the SOA solution stack.

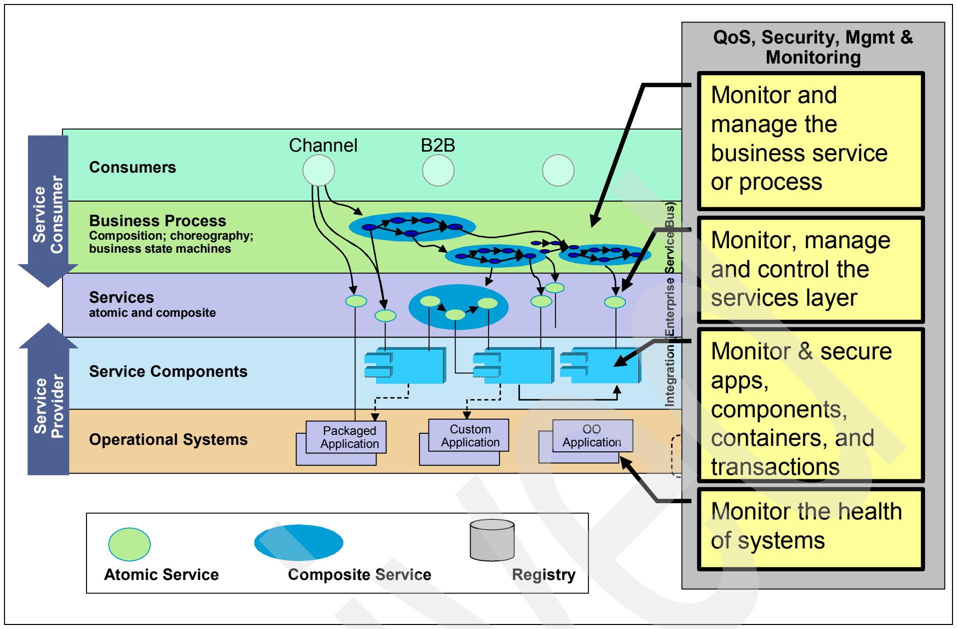

QoS, Security, Mgmt & Monitoring
Monitor and manage the business service or process
Monitor, manage and control the services layer
Monitor & secure apps, components, containers, and transactions
Monitor the health of systems
Service Consumer
Service Provider
Consumers
Channel
B2B
Business Process
Composition; choreography; business state machines
Services
atomic and composite
Service Components
Operational Systems
Packaged Application
Custom Application
OO Application
Integration (Enterprise Service Bus)
Atomic Service
Composite Service
Registry

Figure 7-2 Monitoring in the SOA solution stack

7.2.1 Key architectural decisions with respect to availability management

The following list leads you through the key architectural decisions to be made with respect to availability management:

- Determine the availability requirements for the SOA infrastructure.

  This decision addresses the translation of business user and IT stakeholder requirements into quantifiable availability terms, conditions, and targets, and then into specific availability requirements.

- Define the availability targets and related measures.

  This decision includes the negotiation of achievable availability targets with the business, based on business needs and priorities balanced with current IT capabilities and capacity. Both business application and IT infrastructure elements should be taken into consideration as targets are set.

- Identify the monitoring, analysis, and reporting mechanism.

  This decision includes the continuous monitoring and analysis of operational results data and comparison with service achievement reporting to identify availability trends and issues. Investigate unavailability and identify the underlying causes of problems that have resulted in significant service unavailability.

7.2.2 An availability and performance monitoring solution

In this section, we use a sample IT infrastructure setup to illustrate the availability and performance monitoring solution. This sample configuration consists of application and business services implemented by several major middleware components, such as CICS, IMS, DB2, MQ, and WebSphere Application Server, and network and operating systems. See Figure 7-3 on page 351.

In this sample infrastructure, the CICS and IMS enterprise business applications are running in a z/OS Parallel Sysplex environment. The enterprise database is a DB2 system with DB2 Data Sharing enabled. Business Process services using WebSphere Process Server (WPS) are running on the Linux on System z servers. There are also Web services applications running in WebSphere Application Server servers. The Portal applications for the user interaction are deployed to WebSphere Portal (WP) servers, which are also running on the Linux on System z servers. All WebSphere Application Server, WP, and WPS servers are clustered. The ESB is implemented with WebSphere Message Broker (WMB) servers on Linux on System z, which span across two z/VM LPARs. In addition, a pair of IBM HTTP servers with a WebSphere Application Server plugin and a pair of Edge Servers for IP traffic balancing are also part of this configuration.

This sample scenario shows a very high level view of a business solution that consists of most of the building blocks in the SOA reference architecture, such as Interaction services, Business Application services, Process services, ESB, Information services and Infrastructure services. This is not a very complex environment, but we still need to ensure that the availability and performance requirements are met. In the following subsections, we describe some of the key focus areas for availability and performance monitoring, and map them to the IBM product offerings that can achieve our objectives.

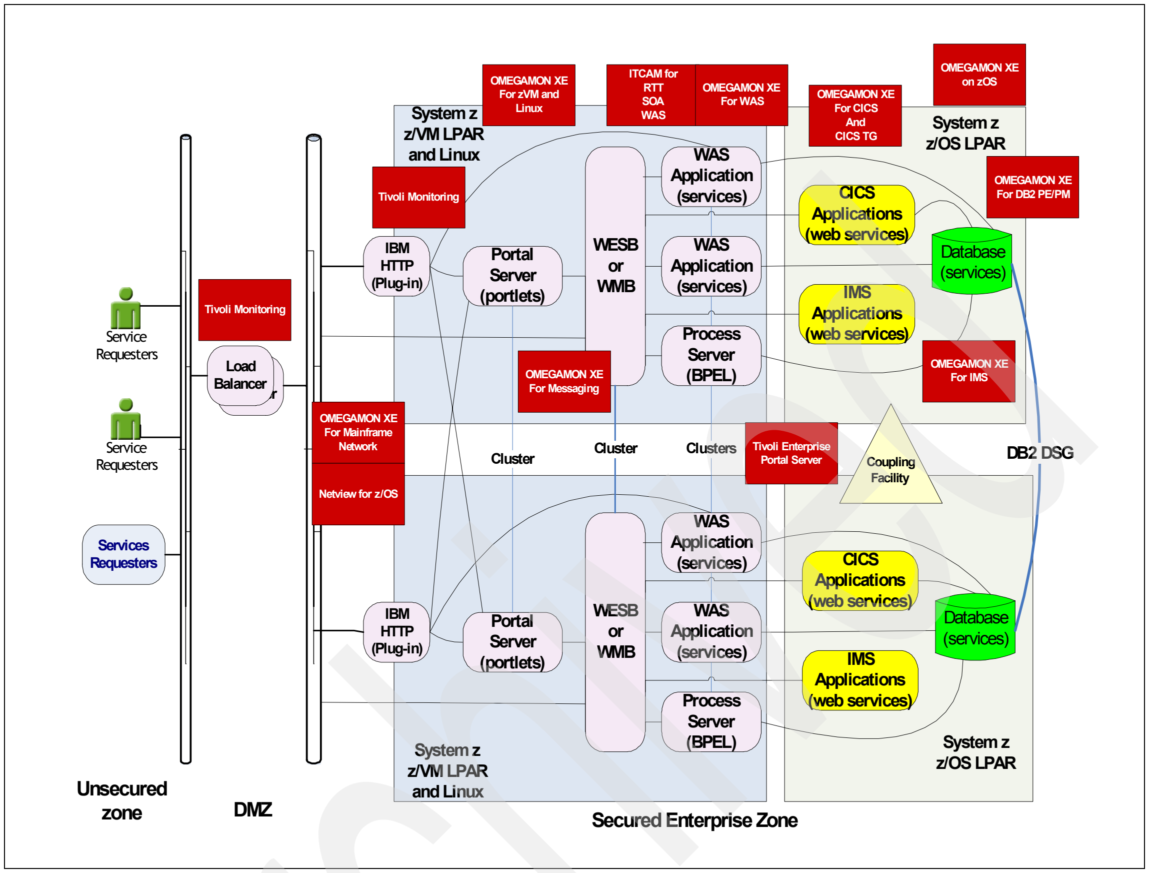


Figure 7-3 Availability and performance monitoring solution

Monitoring Infrastructure services

In this scenario, the Infrastructure services consist of an IBM System z machine, z/OS Parallel Sysplex LPARs, z/VM LPARs and Linux on System z servers. Although all the networking details, such as switches, routers, and firewalls are not shown in the Figure 7-3, they are part of the Infrastructure services and we need to cover them with respect to the availability and performance monitoring solution. We use the following IBM products to achieve our goals:

- IBM Tivoli Monitoring for Linux on System z agent
- IBM Tivoli OMEGAMON® XE on z/OS
- IBM Tivoli OMEGAMON XE on z/VM and Linux
- IBM Tivoli NetView® for z/OS
- IBM Tivoli OMEGAMON XE for Mainframe Networks

IBM Tivoli Monitoring (ITM) for Linux on System z agent is a system monitoring component that manages the IT infrastructure including operating systems, databases and servers across distributed and host environments through a single customizable workspace portal, *Tivoli Enterprise Portal (TEP)*. TEP provides common, flexible, and easy to use browser access and customizable workspaces. We use ITM on all the Linux on System z servers to detect bottlenecks and potential problems in essential system resources and automatically recover from critical situations. ITM offers lightweight and scalable architecture and an easy to use warehouse with advanced reporting capability. When combined with other IBM Tivoli composite application, event, network, and service-level management solutions, it ensures IT resources and staff are operating efficiently and effectively.

IBM Tivoli OMEGAMON XE on z/OS provides a performance and availability monitor for z/OS and OS/390® environments. We use it to validate whether the critical workloads are meeting performance goals. The *z/OS WLM Service Class Resources* workspace provides insight into CPU usage percentage, I/O rates, average storage, and workload goal statistics. We can view the WLM service-class resources to make the necessary adjustments and enable the most important workloads to achieve their performance objectives. The Bottleneck analysis report pinpoints excessive wait times, while impact analysis shows where competing resources are causing delays so we can resolve issues swiftly. The Bottleneck analysis measures wait times for key mainframe resources such as CPU, LPARs, I/O, enqueue, page-in, and many others, indicating where workloads are being delayed.

IBM Tivoli OMEGAMON XE on z/VM and Linux manages performance and availability of z/VM and Linux on System z. We use it to monitor important z/VM and Linux characteristics. It also monitors workloads for the Linux virtual machines, groups, response time, and LPAR information. We use it to identify conditions that could cause outages or performance issues. It also provides all the z/VM resources information including CPU, Paging, Storage, and Workload activities. On the Linux side, it complements data collection by the ITM for Linux on System z agent, for example, it monitors the Linux workloads to detect runaway processes and resource consumption. In addition, it collects and analyzes Linux-specific information such as CPU performance, disk information and performance, network statistics, process status analysis, process user information, system statistics, user login information, and virtual memory statistics.

IBM Tivoli OMEGAMON XE for Mainframe Networks (MFN) proactively monitors and manages the mainframe network components. We use it to show detailed information about application connections, ports, packet size, and volume to identify where network traffic, potential bottlenecks, and other issues consume excessive resources and hamper TCP/IP performance. It generates alerts for resource problems and unstable connections.

IBM Tivoli NetView for z/OS offers an extensive set of tools for managing and maintaining complex, multi-vendor, multi-platform networks and systems from a single point of control. It improves network efficiency and increases system availability. It complements data collection by the OMEGAMON XE for MFN and expands TCP/IP management capabilities with additional support for IPv6, additional active IP connections data, monitoring of hung listeners, simplified processing of SNMP traps, and more granular packet trace formatting and control. It also provides additional resource and path information for SNA-over-IP sessions using Enterprise Extender. Its integration with the *Tivoli Enterprise Portal (TEP)* through the use of a z/OS-based Enterprise Management Agent, enables new and expanded workspaces and views, out-of-the-box situations and expert advice, and broader cross-product integration with the OMEGAMON XE product suite.

Monitoring Business Application services

In this scenario, Business Application services components include CICS and IMS applications and WebSphere Application Server applications. In this sample scenario, CICS applications are exposed as Web services via the CICS Transaction Gateway and CICS Web Services support. IMS applications are also exposed as Web services via the IMS SOAP gateway. All WebSphere Application Server Web services applications are running on WebSphere Application Server servers on Linux on System z. However, WebSphere Application Server Web services applications can also run on the z/OS LPARs. We need to ensure the availability and performance goals are met. We use the following IBM products to achieve our goals:

- IBM Tivoli OMEGAMON XE for CICS on z/OS
- IBM Tivoli OMEGAMON XE for CICS TG on z/OS
- IBM Tivoli OMEGAMON XE for IMS on z/OS
- IBM Tivoli OMEGAMON XE for WebSphere Application Server distributed systems
- IBM Tivoli Composite Application Manager (ITCAM) for WebSphere Application Server
- IBM Tivoli Composite Application Manager (ITCAM) for RTT
- IBM Tivoli Composite Application Manager (ITCAM) for SOA

IBM Tivoli OMEGAMON XE for CICS on z/OS allows us to monitor and manage CICS transactions and resources. We can quickly detect and isolate problems when they occur in the complex CICS systems environment to minimize or eliminate any impact to the business. We monitor transactions and manage how resource usage and performance affect the business systems.

IBM Tivoli OMEGAMON XE for CICS TG on z/OS automatically discovers all active CICS Transaction Gateway regions, and provides a quick and easy way to monitor and manage the CICS Transaction Gateway on z/OS environment. Individual CICS TS Server Analysis pertinent to CICS Transaction Gateways allows the user to quickly identify the CTG transaction and associated CICS TS server. CTG workload and resource utilization views provide comprehensive insight into the CTG environment to ensure critical workload is running efficiently.

IBM Tivoli OMEGAMON XE for IMS on z/OS provides a tool to optimize performance and availability of vital IMS systems. It reduces potential delays or outages by reporting Coupling Facility (CF) structure statistics, shared queue counts, database lock conflicts, and other critical IMS attributes. It provides easy navigation between Tivoli Enterprise Portal (TEP) workspaces and enhanced integration with OMEGAMON XE for DB2 and OMEGAMON XE for CICS. It offers rich monitoring of IMS Connect TCP/IP transaction request time, internal response time, and total response time. It detects and prevents application bottlenecks and response time problems.

IBM Tivoli OMEGAMON XE for WebSphere Application Server on distributed systems (and on z/OS) agents provide availability and performance management capabilities for WebSphere Application Server on distributed systems and z/OS. We use it to track bottlenecks down to the method and API call levels. The workload analysis feature provides an internal snapshot of the application and instantly isolates problems to a specific component. Advanced memory leak detection features help to pinpoint leaks to a specific line of code or show you when two areas of code compete for the same resources. OMEGAMON XE for WebSphere Application Server captures more than 800 metrics to provide a detailed picture of the application server environment.

IBM Tivoli Composite Application Manager (ITCAM) for WebSphere Application Server agents provide real-time problem detection, analysis, and repair to help maintain the availability and performance of on demand applications. ITCAM focuses on the composite application and provides a deep view of all J2EE transactions that are "in-flight" to uncover the root cause of bottlenecks and perform detailed memory analysis. It correlates and profiles transactions that span multiple subsystems, for example, transactions spanning J2EE, CICS, and IMS can be correlated with the optional CICS and IMS data collectors. Resource or application traps can be set to detect and remedy potentially troublesome situations.

IBM Tivoli Composite Application Manager (ITCAM) for Response Time Tracking (RTT) agents monitor end-user response time and help visualize the transaction's path through application systems including CICS and IMS, including response-time contributions of each step. ITCAM for RTT proactively recognizes, isolates, and resolves transaction performance problems using robotic and real-time techniques. It drills down each of the transaction's steps across multiple systems and measure each transaction component's contribution to overall response time. ITCAM for RTT offers new tracking support for business processes written in BPEL running in WebSphere Process Server and support for Portals in WebSphere Portal Server.

IBM Tivoli Composite Application Manager (ITCAM) for SOA agents monitor, manage, and control SOA services for high availability and performance. ITCAM for SOA speeds up and simplifies identification and resolution of SOA problems through a services topology view that provides actual views into service flows by displaying service-to-service relationships, including drill down to service status and metrics. It automates SOA management and helps meet established service levels through built-in alerts, message mediations, situations, and workflows. ITCAM SOA leverages smooth integration with other IBM Tivoli and WebSphere products to provide a comprehensive application management solution for complex environments.

Note: At the time of writing, the ITCAM management server is not supported on z/OS. However, it can be run on distributed systems including Linux on System z.

Figure 7-4 on page 356 shows a sample scenario of monitoring Business Application services.

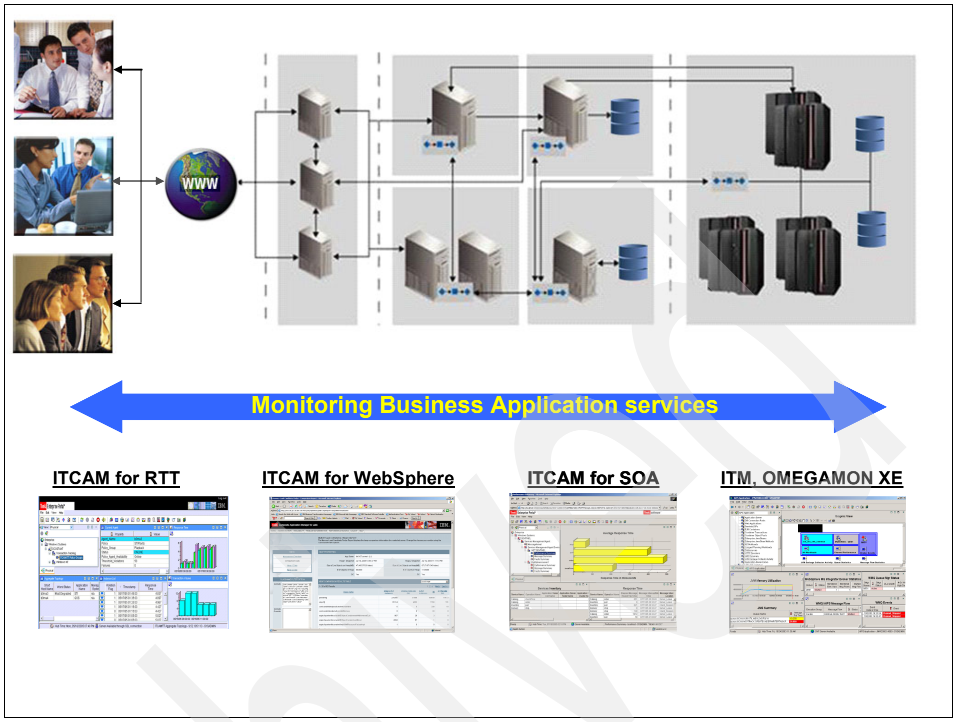


Figure 7-4 Monitoring Business Application services

For the transactions dimension, ITCAM for RTT provides end-to-end transaction tracking to quickly identify and isolate problems.

For the applications dimension, ITCAM for WebSphere Application Server provides drill-down diagnostics for application performance problems for J2EE, CICS, MQ, and IMS, as well as providing data and analysis for the operational view of all resources.

For those applications based on SOA, ITCAM for SOA provides both operational mediations and deep dive information about the critical services supporting SOA-based applications.

For the resource monitoring dimension, IBM Tivoli OMEGAMON XE for Messaging provides resource analysis for WebSphere MQ and WebSphere Message Broker.

Monitoring Process services

In this scenario, Process services components include the IBM WebSphere Process Server (WPS). In this sample scenario, Web services are being orchestrated by the *Business Process Engine (BPE)*. The Web services being integrated by the Business Process services are running in the WebSphere Application Server servers on Linux on System z and in CICS on z/OS LPARs as Web services. We need to ensure the availability and performance goals are met. We use the following IBM products to achieve our goals:

- IBM Tivoli OMEGAMON XE for CICS on z/OS
- IBM Tivoli OMEGAMON XE for WebSphere Application Server distributed systems
- IBM Tivoli Composite Application Manager (ITCAM) for RTT
- IBM Tivoli Composite Application Manager (ITCAM) for SOA
- IBM Tivoli OMEGAMON XE for Messaging for distributed systems (and z/OS)

The usage of the monitoring products is very similar to the monitoring of Business Application services, so we do not repeat the details here. In general, we use IBM Tivoli OMEGAMON XE for CICS on z/OS to track the CICS Web services application on z/OS. We use IBM Tivoli OMEGAMON XE for WebSphere Application Server distributed systems (or z/OS) to monitor the underlying WebSphere Application Server runtime for WPS. We use ITCAM for RTT and ITCAM for SOA to monitor the availability and the performance of the BPEL processes and the Web services. In this scenario, we implemented the ESB using WebSphere Message Broker (WMB) on Linux on System z, and therefore, we also use IBM Tivoli OMEGAMON XE for Messaging for distributed systems to track the activities of WMB on Linux on System z. However, in case WMB is implemented on z/OS, we could use IBM Tivoli OMEGAMON XE for Messaging for z/OS to track the activities in WMB on z/OS.

Monitoring Information services

In this scenario, the enterprise data is residing on the z/OS LPARs using DB2. We use another IBM product to monitor and manage this environment.

IBM Tivoli OMEGAMON XE for DB2 Performance Monitor (PM) monitors individual data-sharing members or entire data-sharing groups. We use it to monitor applications running in a parallel query environment, even if the parallel tasks are executed on different processors. It provides near-term performance history to see problems that otherwise go unnoticed and prevents them in the future. It provides object analysis of database disks, tables, table spaces, and other elements to tune performance.

In addition, we also use *IBM Tivoli OMEGAMON XE for DB2 Performance Expert (PE)*, and combined with the OMEGAMON XE for DB2 PM and DB2 buffer pool analyzer features, we have comprehensive management capabilities for DB2 on z/OS. DB2 OMEGAMON XE for DB2 PE provides a real-time monitoring and enterprise-wide systems management capability. We can drill down to locate problems, identify root cause and resolve bottlenecks or outages effectively. We can automate the analysis of the database performance in real time, store performance data, and run analysis tools in a performance warehouse. We monitor, analyze, and tune the performance of the DB2 database. DB2 PE can provide us with information about current buffer pool behavior and we can use it to simulate and anticipate future behavior.

Monitoring ESB services

The ESB is the core and the major component in this scenario. We do not repeat the discussion about ESB roles and responsibilities in SOA here (refer to Chapter 2, “Enterprise Service Bus services” on page 5). Because it is a key component of this sample integration solution, we need to ensure its availability and performance goals are met in full.

IBM Tivoli OMEGAMON XE for Messaging for distributed systems (and z/OS) proactively monitors and manages WebSphere MQ and WebSphere Message Broker. We use it to identify problems and automate corrective actions using predefined industry best practices while monitoring key WebSphere MQ and WebSphere Message Broker metrics. It provides us with comprehensive monitoring using predefined workspaces providing statistical information such as Current Message Rates, Current Average Message Time, and Sub-flow Statistics. We can define, verify, and schedule automatic deployment of WebSphere MQ configurations enterprise wide with centralized configuration management for complex WebSphere MQ, WebSphere Message Broker environments. In addition, it allows us to proactively prevent problems by detecting and repairing problems as they happen, or sending alerts.

Monitoring Interaction services

Interaction services provides the entrance to this sample business application scenario. The Portal server is used to deploy the Portal applications. Therefore, we also require a solution to ensure its availability and that the performance goals are met, too. We use the following IBM products to achieve our goals:

- IBM Tivoli OMEGAMON XE for WebSphere Application Server distributed systems
- IBM Tivoli Composite Application Manager (ITCAM) for RTT

The usage of the monitoring products is very similar to the monitoring of Business Application services and monitoring of Process services; therefore, we do not repeat the details. In general, we use IBM Tivoli OMEGAMON XE for WebSphere Application Server distributed systems to monitor the health of the underlying WebSphere Application Server run time for WebSphere Portal Server. We use ITCAM for RTT to track the end-to-end response time.

7.2.3 System automation solution

In the complex IT environment, we need to mitigate downtime due to unanticipated growth. We need to maintain service levels when new applications are implemented. We also require easy management of complex infrastructures with advanced, policy-based automation in order to accommodate future growth and handle varying levels of demand with a scalable solution. Therefore, a robust system automation solution to coordinate activities using a single point of control to help ensure enterprise-wide availability is significantly important. The solution should help reduce the cost of downtime with improved availability and operational continuity. Among the key criteria for this solution are:

- Provide end-to-end automation solutions with the self-detecting and auto-correcting capabilities
- Able to automatically react to events that threaten infrastructure performance and availability
- Eliminate the need for error-prone manual intervention, thereby helping to improve availability, lower costs, and speed the time to recovery

IBM Tivoli System Automation for z/OS (SA z/OS) is the only host automation solution based on IBM Tivoli NetView for z/OS. It is part of the IBM Tivoli System Automation family of products. Together with IBM Tivoli System Automation for Multiplatforms, it provides high availability for the critical applications and IT services running on the distributed systems, including Linux on System z. Therefore, it can be easily integrated and provide high availability of multi-tier business applications that span across the heterogeneous IT environment and allow us to more effectively automate the composite applications and increase availability.

IBM Tivoli System Automation for z/OS provides the following benefits:

- Proactively manage availability and performance through performance-driven automation
- Integrate z/OS into end-to-end automation of complex applications spanning multiple platforms using IBM Tivoli System Automation for Multiplatforms
- Integrated with IBM Tivoli Workload Scheduler (TWS), thereby allowing a TWS user to issue any SA/NetView command as part of a TWS operation
- Provides advanced IBM Geographically Dispersed Parallel Sysplex™ (GDPS) capabilities and integration
- Provides management and automation covering CICS, IMS, DB2, WebSphere Application Server, and z/VM
- Automate processor and system operations and I/O resources, including channels, devices, ESCON® and FICON Directors

7.2.4 A workload management solution

In the SOA environment, we move from a static, platform-based view of production workloads to a flexible, service-driven environment. We need a workload management solution so that we can centrally manage and automate application workloads across heterogeneous environments. Key criteria that the workload management solution should meet include the capability to:

- Automate, plan and control the processing of production workload.
- Drive workload performance according to business objectives.
- Perform dynamic real-time workload automation in addition to calendar and event-based scheduling.
- Automatically process heterogeneous workloads according to business policies, to minimize idle time and improve throughput.
- Monitor and manage workloads by exception, and create production run time reports.
- Trigger batch job and job stream execution based on real-time events and notify users when unusual conditions occur in the infrastructure or batch scheduling activity.
- Workload management with real-time alerts and user notifications, self-monitoring, self-healing, automated recovery, and user-defined event rules for batch job scheduling.
- Do enterprise resource planning (ERP) workloads and extend capabilities with easy integration to third-party products.

IBM Tivoli Workload Scheduler (TWS) for z/OS leverages service-oriented architecture to align workload automation to business processes, and integrate with other workload schedulers and systems management products. It can automate, plan, and control the processing of entire enterprise-wide production workloads, not just the batch subset. The Tivoli Workload Scheduler for z/OS provides the following benefits:

- Provides event-driven scheduling and the ability to manage real-time event-triggered workloads in addition to calendar-triggered workloads.
- Integrated with TWS for the distributed systems and provides the end-to-end workload automation.
- Monitors System z or enterprise-wide workloads with the IBM Tivoli Dynamic Workload Console shipped with TWS for z/OS.
- Performs critical path analysis and critical workload management.
- Integrated with z/OS Workload Manager (WLM) and dynamically routes the workloads to best available System z resources.

- Monitors System z resources by integrating with IBM Tivoli System Automation for z/OS.
- Integrated with Tivoli Enterprise Portal (TEP), thus job events and status can be monitored.

Figure 7-5 shows an overview of the TWS architecture.

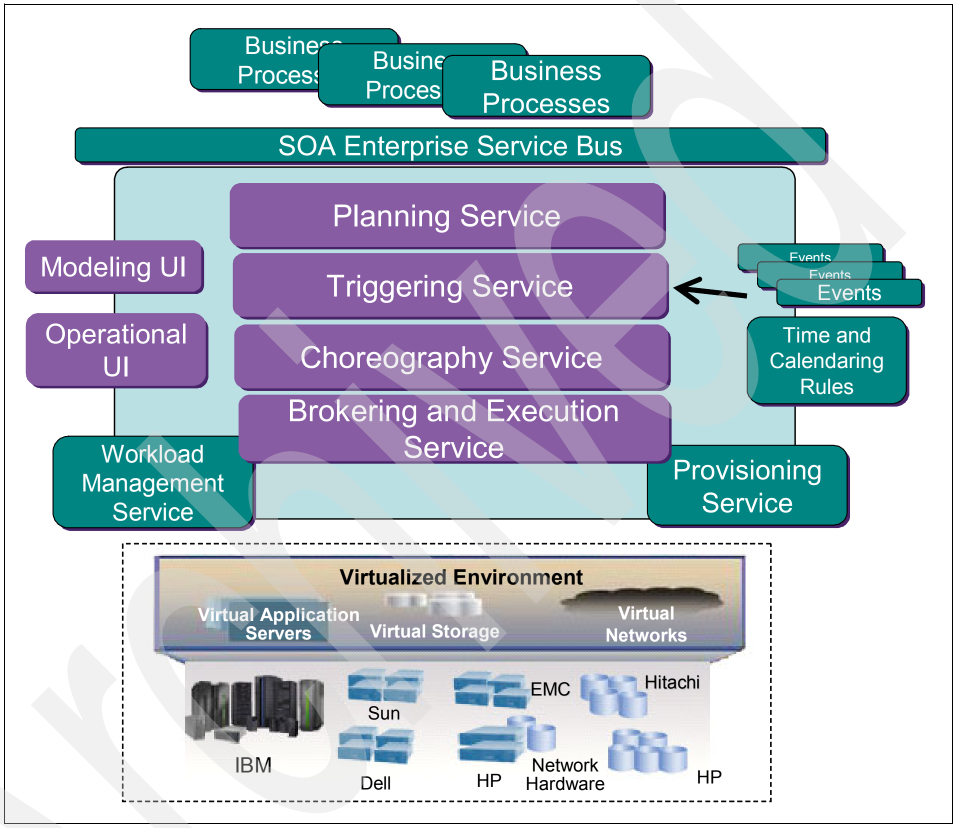


Figure 7-5 TWS architecture in an SOA context

IBM Tivoli Workload Scheduler for z/OS is a key element in a comprehensive, on demand, Tivoli Dynamic Workload Automation Family Portfolio.

According to business needs or organizational structure, distributed and z/OS components can be used in a mix of configurations for a completely distributed workload automation environment, a completely z/OS environment, or a mixed “End-to-End” z/OS and distributed environment. The *Tivoli Workload Scheduler for Applications* component extends Tivoli Workload Scheduler to automate workloads on ERP systems and non-native platforms.

IBM Tivoli Dynamic Workload Console (TDWC) provides a single point of control for System z workloads or enterprise-wide workloads in end-to-end environments. *Tivoli Dynamic Workload Console (TDWC)* is the new Web-based GUI for the TWS portfolio. It is a component of TWS for z/OS and is included and fully supported at no extra charge.

TWS also can be integrated with other SOA Service Management solutions such as Business Service Manager, Service Level Advisor, and so forth, providing a complete workload management solution.

Figure 7-6 shows the workload management solution using TWS.

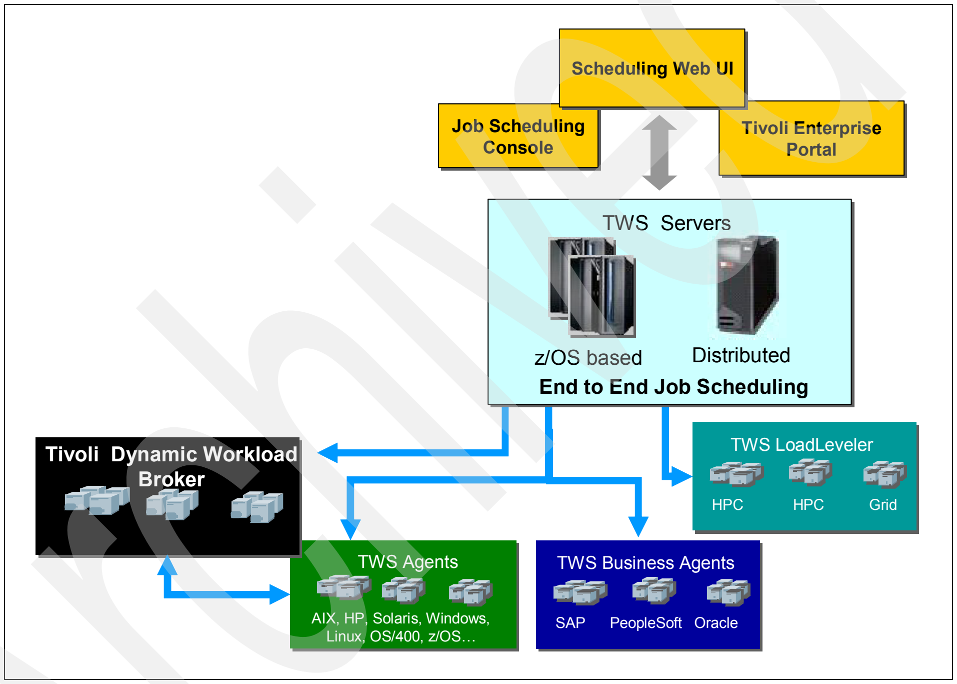


Figure 7-6 Workload management solution using TWS

7.3 Security management

Securing access to information is important to any business. Security becomes even more critical for implementations structured according to SOA principles due to loose coupling of services and applications and their possible operations across trust boundaries.

Because System z usually possesses assets that are critical for the enterprise, it becomes even more important to enforce best security practices for System z SOA services.

7.3.1 IBM SOA Security Reference Model

Along with the creation of all the services composing a service-oriented architecture, there is a need to simplify, enhance, and adapt their security. For this purpose, there is a need for "security as a service." Security as a service allows building security capabilities and rules as services in order to benefit from centralized management and reuse. All aspects of security are concerned in the transformation to security services.

A reference model can help to address requirements and lead to a logical architecture, then to a physical architecture, with products and technologies mapped to solve the current needs.

The *IBM SOA Security Reference Model*, shown in Figure 7-7, is a sub-component of the IT Service Management pillar.

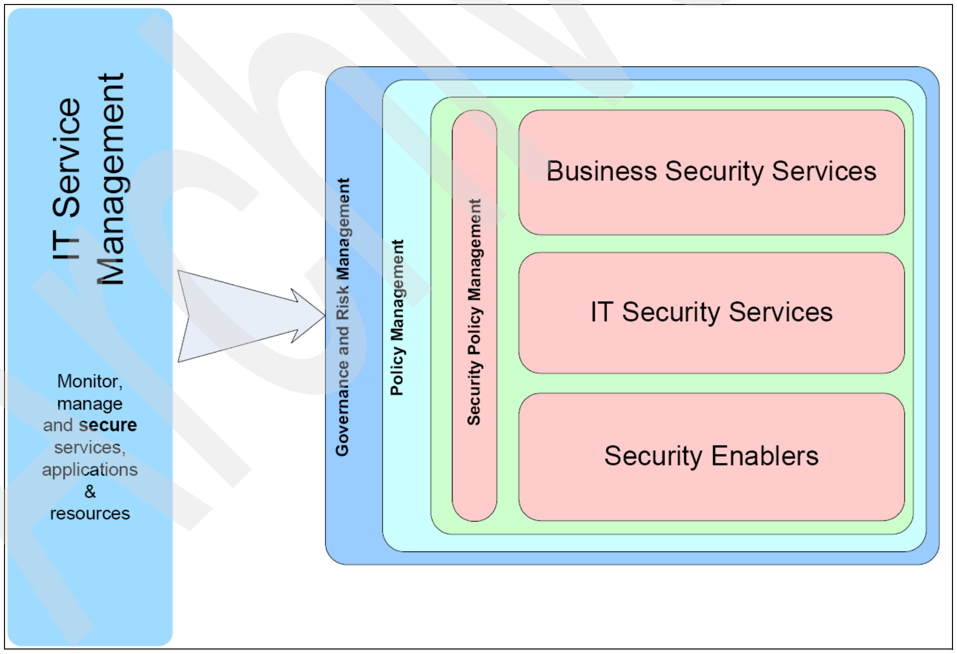

IT Service Management
Monitor, manage and secure services, applications & resources
Governance and Risk Management
Policy Management
Security Policy Management
Business Security Services
IT Security Services
Security Enablers

Figure 7-7 IBM SOA Security Reference Model: Security capability within IT Service Management

The reference model can be viewed in terms of three layers of abstraction: Business Security services, IT Security services, and Security Policy management. There are also Security enablers for providing security functions to the IT Security services.

At a very high level, we can define the following main areas in the model:

- *Business Security services* involve managing the needs and requirements of the business, such as trust, identity and access management, data protection, compliance and reporting, non-repudiation, and secure systems and networks. They build on the common policy infrastructure to effectively manage the relevant policies applicable to meet the business needs.
- *IT Security services* is a set of consistent security services that can be used by different components of an SOA run time. This set of services is flexible and allows for different mechanisms to plug in, such as user registries or authorization engines. It enables a consistent security implementation. It also minimizes development and deployment costs for these security services and for the SOA environment on which these security services are reused.
- *Security enablers* are utilized by the IT Security services to perform their task. Examples include technologies such as cryptography, registries and repositories, key management, malware protection, isolation, firewalls, intrusion detection and prevention, and so on.
- A *policy driven* approach is fundamental to the success of SOA. The goals established and driven by the business must be implemented and enforced by the infrastructure. To achieve this, a complete policy management framework must be in place. There are three aspects to policy: the abstraction level of policies, life cycle management for policies, and the domains to which policies can be applied. Security is an important part of the overall policy management.
- *Security Policy management* in the context of SOA is the means by which processes and services express the conditions for their use, and manage the behavior of the underlying infrastructure in order to secure access to information, provide confidentiality and integrity, enable audit, and so on.
- *Governance* of SOA Security is a subset of the overall SOA Governance function. Governance is very important for the security services because managing the security policy and implementation is vital to the integrity of the environment. A framework for the effective governance structure and decision making authority is needed in order to run the business. Tools and technologies can help facilitate governance initiatives and compliance evaluations. An effective security governance framework involves establishing chains of responsibility, authority, and communication to empower people to effectively control the system.

- *Risk management* deals with the process of evaluating and assessing risks in the SOA environment and developing strategies to manage those risks. Risk management balances risks with the cost of managing them. The risk management process determines how to manage risk based on factors such as probability and impact.

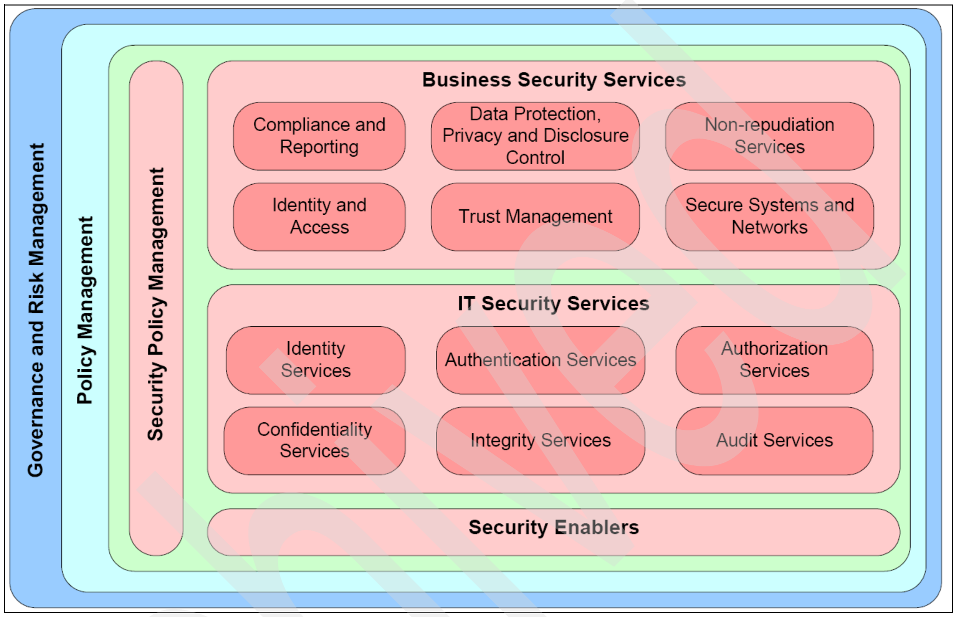


Figure 7-8 IBM SOA Security Reference Model

Business Security services

Referring to Figure 7-8, Business Security services include the following services:

- *Compliance management* measures the performance of the business/IT system relative to the measures established by the business controls and policies. These can be realized based on reporting on system behavior using audit information (about all events, and not limited to security events), and comparing that behavior to configured policies in systems.
- *Data protection management* deals with protecting business information and provides the capabilities for data protection in transit and at rest. It includes policies for which data is to be protected and to what extent it can be specified and implemented.
- *Non-repudiation services* allow for protection of the requestor and the provider from false denials that data has been sent or received. Non-repudiation services provide proof of data origin and delivery. They aim

to prevent parties in a communication from falsely denying having taken part in that communication.

- *Identity and Access* manages identity life cycle including provisioning and de-provisioning identities and self-care/self-registration for optimal user interaction. Identity and Access also includes processes and policies for approval of access to IT resources and business resources.
- *Trust management* addresses trusted relationships between entities, such as organizations, enterprises, identities, security domains, and systems. These relationships can be system-to-system, business-to-business, and so on.
- Business policies are required for intrusion detection and event management for ensuring *Secure systems and networks*. Processes must be in place for handling alerts and for normal housekeeping of scheduled maintenance, patch management, and servicing. This is a category of technologies and embedded systems that help protect infrastructure servers, systems, and networking resources from security threats. The goal is to protect the systems from external and internal threats, such as malicious hackers and malware.

IT Security services

Referring again to Figure 7-8, IT Security Services include the following services:

- *Identity services* are an abstraction layer and framework that provide for managing, sharing, federating, and accessing identity information from a variety of authoritative identity sources. These services also manage relationships between identities and provide identity information to multiple identity systems. The identity services span from a simple directory service for user repositories to complete identity life cycle management with user provisioning.
- *Authentication services* provide capabilities to authenticate an identity. Authentication is the process of validating the identity claimed by the accessing entity. These services can support multiple authentication mechanisms, such as user name/password, hardware token-based, or biometric-based. They can also support protocols, such as Kerberos.
- *Authorization* follows authentication. That is, once a user or system has been authenticated, it is then possible to perform authorization. Authorization means making a decision about whether an authenticated, or even an unauthenticated identity is allowed to access a resource. An authorization decision depends on an authorization policy and an authenticated user or system.
- *Confidentiality services* are the security services for ensuring non-disclosure of sensitive information travelling through untrusted communication networks or at rest, such as in data stores, volatile memory, and so on.

- *Integrity services* are security services for detecting unauthorized modification of data due to errors or malicious attacks.
- *Audit services* include maintaining detailed, secure logs of critical activities in a business environment. These critical activities can be related to security, content management, business transactions, and so on.

More detailed information about the IBM SOA Security Reference Model can be found in *Understanding SOA Security Design and Implementation*, SG24-7310.

7.3.2 Key architectural decisions with respect to security management

In this section, we present a checklist of key architectural decisions to be made with respect to Security services and System z.

When designing the architecture of IT Service Management security services, the following decisions need be made:

- Should security enforcement be externalized out of the application logic and be handled by the infrastructure?
- Should security enforcement be centralized within external common security services?
- Must security policies management be centralized across applications, middleware, and intermediaries?
- Are security requirements different for different access path, for example, from inside or outside of the enterprise?
- Must a defense in-depth approach be used by enforcing security policies at intermediaries, middleware, and applications consistently throughout the organization?
- Do requests cross security domains, and do policies need to be managed across security domains?
- Is there a need for Single sign-on or identity propagation or security attribute propagation?
- Will business partners be allowed to access services from business partners using identities defined in their organization?
- Is there a need for chain of identities in the case where composite service can access multiple other services for completing its tasks?
- Is auditing required in order to meet some compliance goals?
- Is there a need to federate the z/OS security domain or the Linux on System z security domain with distributed security domains?

- Is a central z/OS identity service required?
- Should the z/OS central identity service be available to distributed platforms?
- Is a central identity service with high quality of service needed?
- Is there a need to integrate z/OS identity service with other identity services?
- Is central external authentication needed?
- Is central external authorization needed?
- Are central integrity and confidentiality services needed? If so, which ones?
- What are the auditing and compliance requirements?

7.3.3 Patterns

Patterns described in this section focus on demonstrating the impact of security as a service as well as the integration of heterogeneous security domains.

We discuss the following patterns:

- Central identity services
- Central external authentication services
- Central external authorization services
- Federation of heterogeneous security domains

Central identity services

The Identity services are an abstraction layer and framework that provide for managing, sharing, federating, and accessing identity information from a variety of authoritative identity sources. These services also manage relationships between identities and provision identity information to multiple identity systems.

Because of the nature of software solutions, it is common to have dispersed Identity services among all of the runtime services. For example, a Portal might have its own LDAP rich user registry, an application server might have a different LDAP registry, while a database might use a few generic identities stored in a configuration file. This is also true for other Identity services such as identity life cycle management and identity replication.

Figure 7-9 shows Identity services dispersed and then centralized.

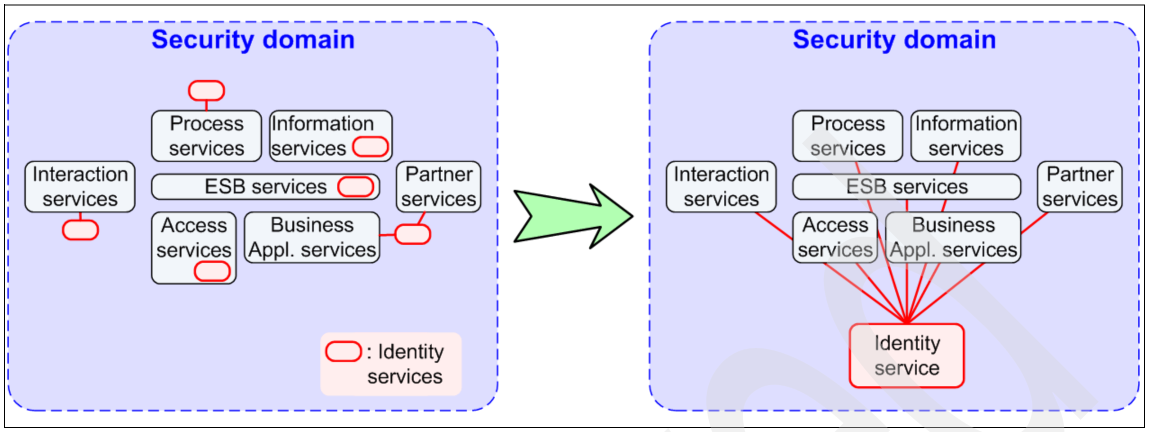


Figure 7-9   Identity services dispersed and centralized

Security as a service helps centralize and reuse Identity services. This simplifies the management of Identity services and also the integration of runtime services with Identity services. Common interfaces can help for this purpose. As an example, z/OS System Authorization Facility (SAF) allows most z/OS sub-systems to access a common user repository. In addition, standards can help a lot for this purpose. As an example, the Lightweight Directory Access Protocol (LDAP) standard facilitates the deployment of centralized user repositories mainly for distributed and Linux on System z platforms, but is also available for some z/OS products.

Central external authentication services

Authentication services provide capabilities to authenticate an identity. Authentication is the process of validating the identity claimed by the accessing entity.

Most of the time, if there is no trust with the outside security domain yet, there is a need to authenticate any client which accesses any runtime service inside the security domain.

Because of the nature of software solutions, it is common to have dispersed authentication services among each of the runtime services. For example, a Portal might have its own form-based login relying on an LDAP user registry. An ESB might authenticate using the WS-Security protocol using its own LDAP user repository. A business Web application might use Basic Authentication with its own user registry.

Figure 7-10 shows dispersed authentication services.

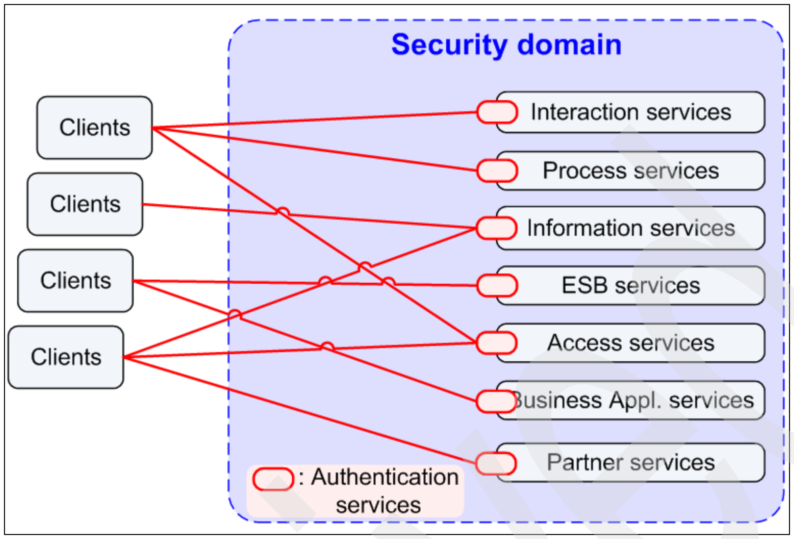


Figure 7-10 Authentication services dispersed

Security as a service helps centralize and reuse Authentication services. This simplifies the management of Authentication services and also the integration of runtime services with Authentication services.

It also reduces the exposure of runtime services inside the security domain because client requests are only allowed to enter through the Authentication services. Having client requests enter through an Authentication service is often enforced with two lines of firewalls, which constitute a *DeMilitarized Zone (DMZ)*. Authentication services are commonly run by *Reverse Proxy Security Servers (RPSS),* which can also have other functions such as some authorization features, some SSL/TLS endpoints capabilities, or some caching services.

Figure 7-11 on page 371 shows centralized authentication services.

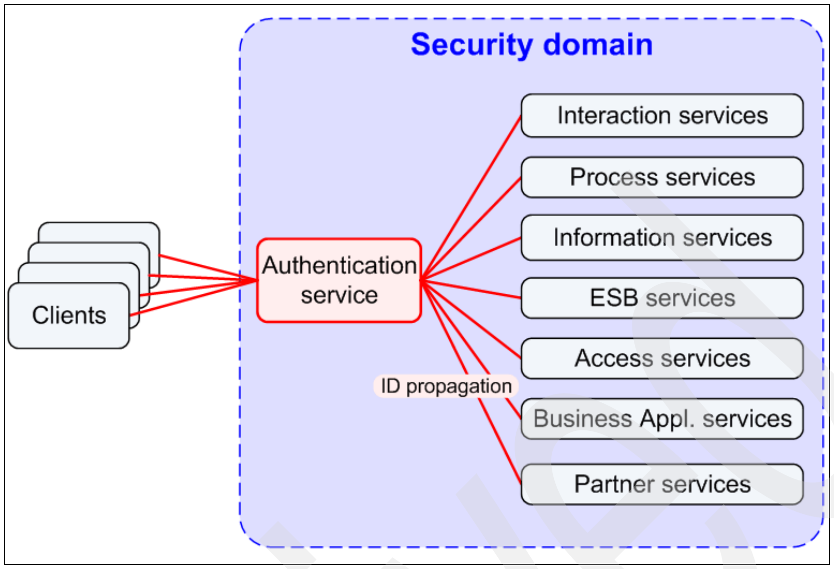


Figure 7-11 Authentication services centralized

Because the Authentication services are external to runtime services and because runtime services might still need to know who is the client owner of a request, it is often necessary to have the client identity flow from the Authentication services to the runtime services. This is called *Identity Propagation* or *Single Sign On*. More advanced security architectures can also use *Security attribute propagation,* where not only the identity but also a more complex security context is propagated. The identity can flow using some transport layer mechanisms such as Basic Authentication HTTP headers or using message layer Username tokens, or advanced binary tokens, or even some standard tokens such as *Security Assertion Markup Language (SAML)* tokens.

Central external authorization services

Authorization is the process of checking whether an asserted (already authenticated) identity has access to a requested resource. Authorization means making a decision about whether an authenticated, or even an unauthenticated identity is allowed to access a resource. An authorization decision depends on an authorization policy and an authenticated user or system.

Because of the nature of software solutions, it is common to have dispersed authorization services among each of the runtime services. For example, a portal might have its own authorization rules stored in its configuration. An ESB might authorize depending on roles stored inside its own configuration. A business application might use authorization rules stored in a specific security server.

Figure 7-12 shows authorization services dispersed and then centralized.

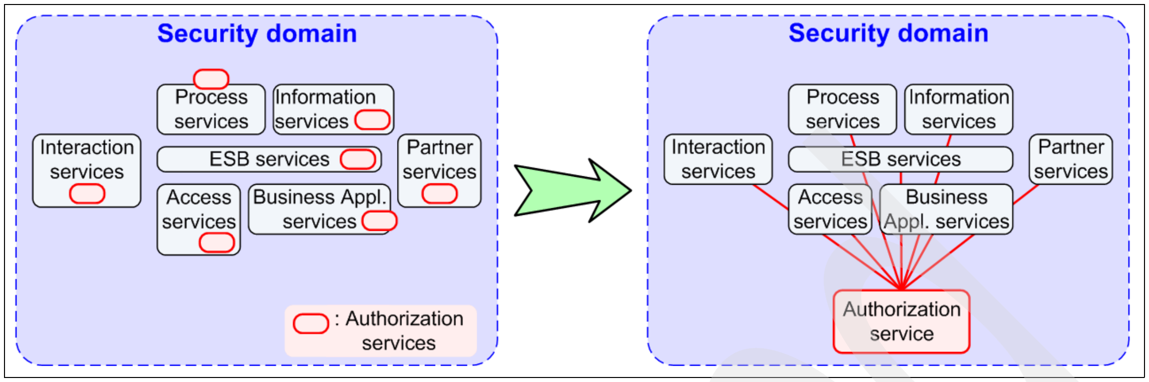


Figure 7-12 Authorization services dispersed and centralized

Security as a service helps centralize and reuse Authorization services. This simplifies the management of Authorization services and also the integration of runtime services with Authorization services. Common interfaces can help for this purpose. As an example, z/OS *System Authorization Facility (SAF)* allows most z/OS sub-systems to access the authorization rules (classes, permissions). Also, standards can help a lot for this purpose. As an example, *Java Authorization Contract for Containers (JACC)* standard facilitates the deployment of centralized authorization providers mainly for distributed and Linux on System z platforms, but is also available for some z/OS products.

Federation of heterogeneous security domains

SOA facilitates the integration and the reuse of services within silos and also across traditional boundaries. Security follows accordingly and has to secure all aspects of these new integration models.

One integration model relevant to System z is the integration of System z security domains with distributed security domains. A security domain is a group of clients and servers under the control of one common set of security services. Most of the time, a security domain has its own set of identities and its own set of authentication and authorization services. z/OS usually has its own security domain with security services being provided for most z/OS sub-systems by the SAF security server. Linux on System z might have its own security domain with some user registry and authentication and authorization services. It is also common to see Linux on System z share the same security domain with other Linux or UNIX systems. Other distributed environments might each have their own security domain. For example, Microsoft infrastructure with the Domain Controller and Active Directory® is a security domain.

Along with the reuse of services across the enterprise and outside of the enterprise, there is a need to have security with the capabilities to enforce identification, authentication, authorization, confidentiality, integrity, and non-repudiation across multiple security domains such as System z security domains and distributed security domains.

The quick short-term way to integrate security across independent security domains is to use point-to-point static security between only the components that need to interact. Because the integration is static, it is common to find generic user IDs and passwords being used for authentication across components. For example, an application server in the distributed security domain might provide a hard-coded generic SAF user ID and password for using a database located in the z/OS security domain. This does not allow knowing which end user or which end client accesses the database.

Figure 7-13 shows multiple security domains with their security services, and static security being used for any communication across the security domain.

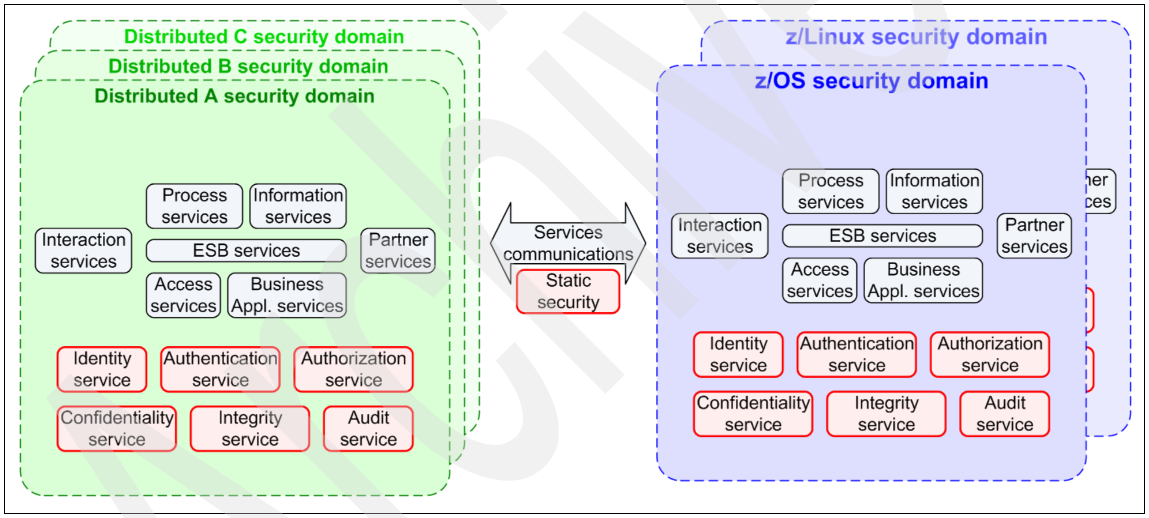


Figure 7-13 Static point-to-point integration of heterogeneous security domains

Static security is good for point-to-point security between components, but static security cannot satisfy the end-to-end security requirements. End-to-end security is about flowing end user or clients' identities across multiple components for finer-grained security or auditing. Thus the security mechanisms that support end-to-end security must be able to change the flowing identity depending on who the end user or end client is.

It is common that identity formats and names do not match between security domains. For example, LDAP identities with X.500 distinguished names do not match 8-character z/OS SAF-RACF identities. So end-to-end security requires some identity transformation using some security mediations.

The security mediation can transform the format of the identity in a request to be understandable by the remote security services. Various formats exist to carry the identity: HTTP Basic Authentication HTTP headers, UsernameToken, BinaryToken, LTPA, SAML, and so on. Also, the security mediation can transform the value of the identity to map the identity from the requester security domain to an identity existing in the provider security domain. There are one-to-one or many-to-one mappings. Many-to-one identity mapping means that many identities from the requestor security domain are mapped to one identity of the provider security domain. One-to-one identity mapping means that each identity from the requestor security domain is mapped to one identity of the provider security domain.

Many-to-one identity mapping has some disadvantages. The main disadvantage is that requests cannot be identified at the user level because a group of users might utilize the same mapped identity. Thus it is not appropriate for fine-grained authorization and for precise auditing. In addition, many-to-one identity mapping needs the establishment of mapping rules which are difficult to maintain and less flexible considering evolving security requirements. On the other hand, it can happen that scalability requirements make it impossible to achieve one-to-one mapping. For example, one user repository implementation might be able to scale up to millions of identities while another user repository cannot. In that case many-to-one identity mapping might be required.

The security mediation can happen on the requester side, on the provider side, or in the middle, within an ESB for example.

The security mediation might need to know the origin and the destination identities at once to do the identity mapping. This might require that some security identity information be transferred or copied from one security domain to the other. Or there could be some automatic provisioning from one security domain to the other in order to facilitate the maintenance. Or some other security functions such as the authorization services might need to transfer some authorization rules from one security domain to the other whenever a new identity is defined. Most security automation across security domains requires the use of adapters. *Security adapters* are used to transfer security information or configuration among security components.

Figure 7-14 on page 375 shows multiple security domains with their security services and security mediations being used for identity or token transformation along with any communication across the security domain. Also this figure shows the adapters that can be used for integrating security information across the security services.

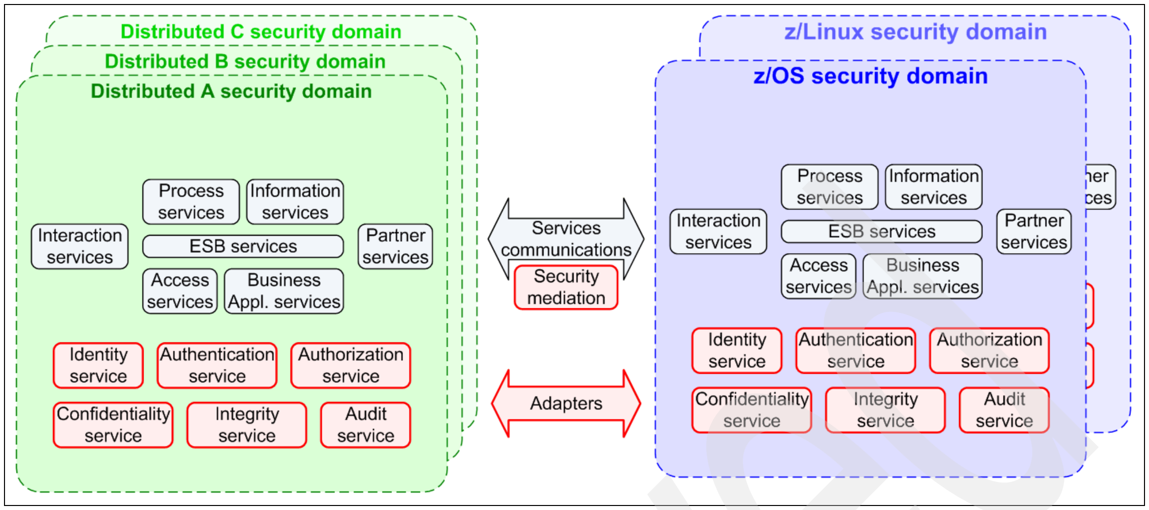


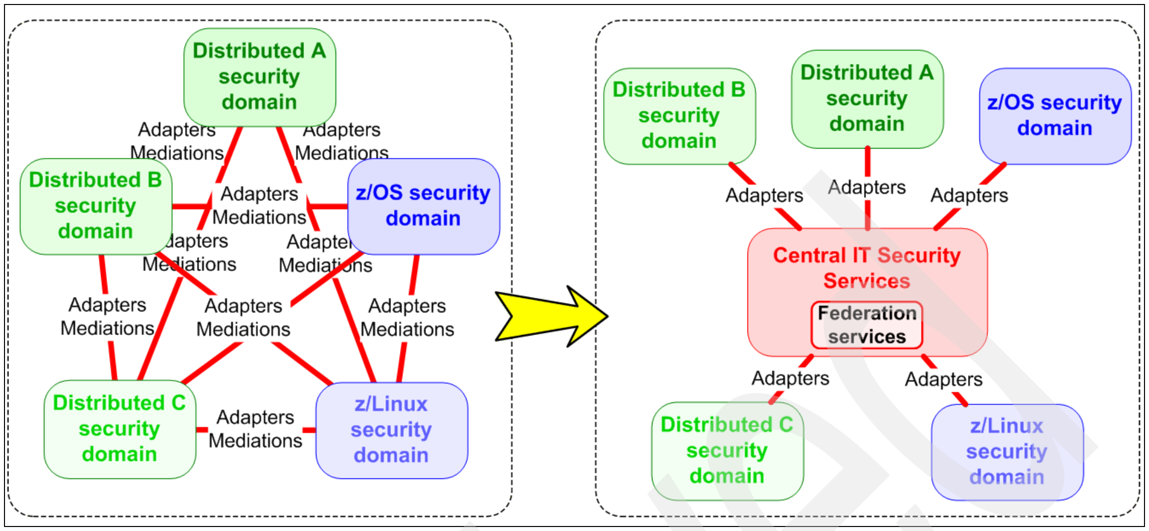
Figure 7-14 Dynamic integration of heterogeneous security domains

Designing, implementing, and maintaining security mediations and adapters across a few security domains is manageable. But with the number of security domains increasing in large corporations and with the necessary integration of all of these within SOA, the number of security mediations and adapters can become massive for all security domains' communication combinations.

One solution to manage the increasing complexity is to have the security architecture rely on some external central IT security services. These central IT security services are external to all existing security domains. Then all security domains contain security information or configuration directed by the central IT security services and sent to security domains using adapters.

When a new security domain is added to the global security architecture, there is no need to define security adapters to all other security domains anymore. When a new security domain is added to the global security architecture, only security adapters to the external central IT security services are necessary.

Figure 7-15 shows the external central IT security services and how it can be beneficial to reduce the complexity of integrating heterogeneous security domains.



Distributed A security domain
Distributed B security domain
z/OS security domain
Distributed C security domain
z/Linux security domain
Adapters Mediations
Central IT Security Services
Federation services
Adapters

Figure 7-15 Central IT Security services and heterogeneous security domains

For example, when a new user gets added to the central Identity directory service, there can be some provisioning to all the necessary Identity directory services for all the security domains that this new user has access to. For ease of management and synchronization, the central Identity directory service can be composed of several identity repository types such as LDAP-based, database-based, and so on. Another example is when a a new user gets added to the central Authorization service, there can be some provisioning of the authorization rules to all the necessary Authorization services for all the security domains that this new user has access to.

With such a security architecture, some security services might not exist locally in the security domain itself, but might only exist in the central IT security services. For example, if the central Identity Directory service contains a master LDAP directory, then it might not make sense to have local replicas for security domains which also need LDAP directories. Then the security domain component can access the central Identity Directory service directly. Another example is a *Public Key Infrastructure (PKI),* which might only exist within the central IT security services in order to simplify the key management.

With such an architecture, all security mediations for end-to-end security across security domains are handled by *Federation services*. Federation services can manage federated single sign-on or identity propagation. Federation services have matured beyond just single sign-on functionality. Standards and specifications such as the SAML specification and WS-Federation and Liberty Alliance ID-FF specifications all now include an aspect of session life cycle management (single sign-on and single sign-off) as well as single sign-on enablement through account linking. This comprehensive approach and enablement of a single sign-on environment is designed to ease the user experience and reduce the cost of management of these users.

Figure 7-16 shows the external central IT security services and the Federation services, which are called during communications across security domains.

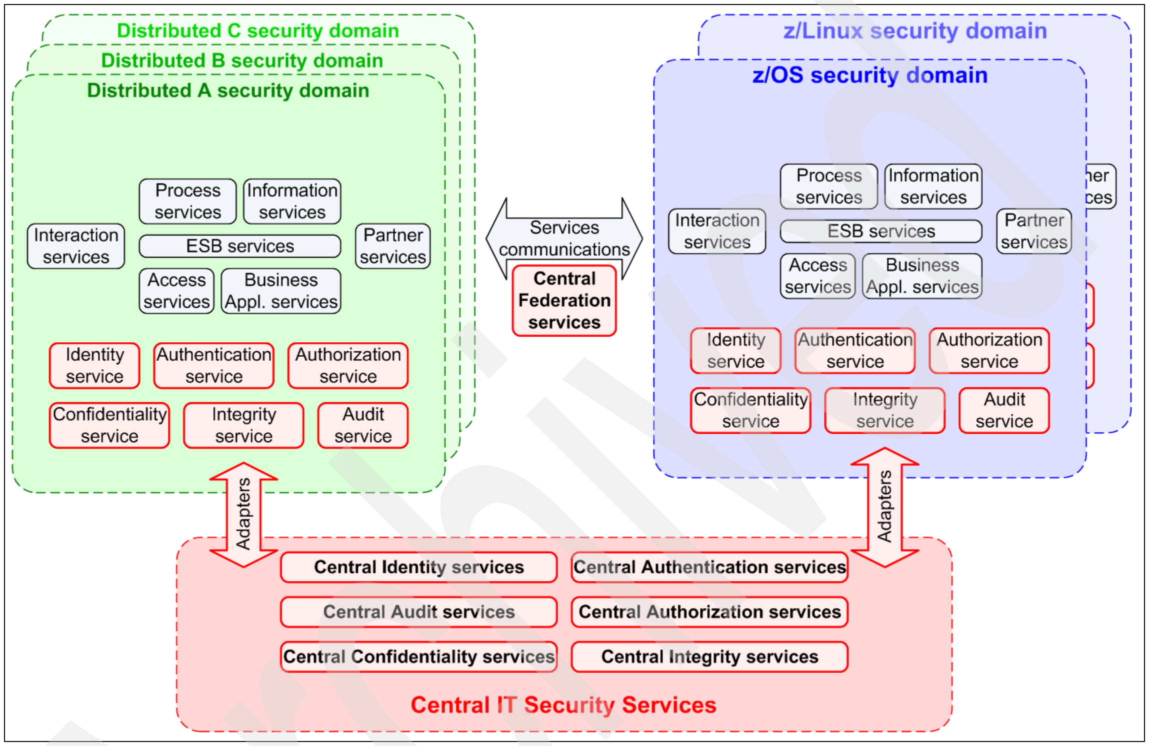


Figure 7-16 Federated heterogeneous security domains

The central IT security services can run on multiple platforms depending on functional and non-functional requirements. Some of them might run partly or completely on System z. For example, it could be a good idea to deploy a central Confidentiality service such as a PKI on z/OS in order to leverage the high security keystores available on z/OS. Also an Identity life cycle and provisioning solution might run on z/OS or Linux on System z to leverage the high security platform for critical security components.

Figure 7-17 on page 378 shows an example scenario of how Identity Federated Single Sign-On or Identity Propagation could be designed with two security domains. The Interaction services could be running on a distributed or a Linux on System z platform. The ESB services are running on both the z/OS and the distributed environments. Whenever a request needs to access Process services or Business Application services deployed on z/OS, the service request would go through the ESB services on distributed, then through the Identity Federated Single Sign On service, then through the ESB services on z/OS, then reach the z/OS Process or Business Application services.

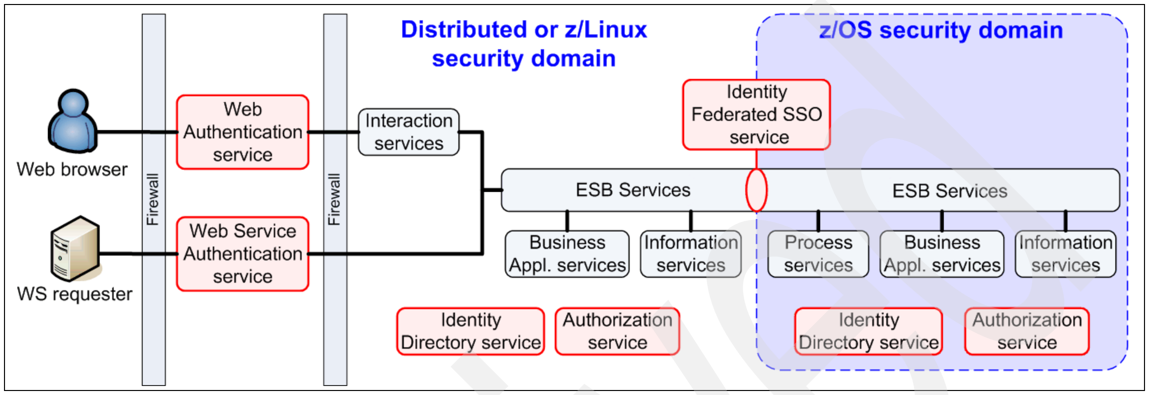


Figure 7-17 Federated heterogeneous security domains scenario

With this scenario, the end user uses a Web browser to authenticate to the Web Authentication service providing a distributed security domain identity. The same is true for an end client that makes a Web service request to authenticate to the Web service Authentication service providing a distributed security domain identity. The end user request first reaches the Interaction services and is authorized considering its distributed security domain identity. Then a request to a z/OS Business Process service would go through the distributed ESB and reach the Identity Federated Single Sign On service, which transforms the distributed security domain identity to a z/OS security domain identity representing the same end user. Then the request arrives at the Business Process services where the appropriate authorization checks and auditing can be done using the z/OS security domain identity. The scenario is similar for a Web service client without the Interaction services step.

7.3.4 Identity services solutions

This section describes the Identity services solutions that exist on System z and how System z can be the trust authority of the central Identity services for the global enterprise architecture.

We discuss the following solutions:

- Central Identity services with SAF-RACF z/OS
- Extend SAF-RACF z/OS Identity services with TDS

- Central Identity services with TDS on z/OS
- Central Identity services with TIM z/OS
- Linux on System z Identity services with TDS
- Linux on System z Identity services with TIM
- Identity services adapter with ITDI

Central Identity services with SAF-RACF z/OS

This solution software runs on z/OS only. This solution can include central Identity services for all z/OS subsystems supporting the SAF interface running on z/OS.

Resource Access Control Facility (RACF) can be a central Identity service for most subsystems running on z/OS. The *System Authorization Facility (SAF)* is part of the operating system and conditionally directs control to RACF.

RACF provides Identity directory services such as storing user identities, associating users to groups, storing passwords or pass phrases, storing user certificates, storing full user names, and so on.

z/OS subsystems such as IMS, CICS, DB2, WebSphere MQ, WebSphere Application Server for z/OS, WebSphere Enterprise Service Bus for z/OS, WebSphere Process Server for z/OS, and many other z/OS products can use RACF as a user registry.

Because all these products can share the identities in this common repository, RACF is an appropriate solution for handling Identity services across a z/OS infrastructure.

Not only is RACF able to be an external central Identity service for all subsystems in one LPAR, but RACF can also be the external central Identity service for all subsystems running on any LPAR in a Parallel Sysplex. For this purpose RACF can rely on a shared RACF database or Remote Sharing Facility.

RACF is designed so that its database can be shared between processor complexes while data integrity is maintained. RACF Parallel Sysplex Data Sharing is an optional function that can facilitate system administration, provide consistent Parallel Sysplex-wide security and improve performance in some environments. Operating RACF in Data Sharing mode requires use of a Coupling Facility.

RACF Parallel Sysplex Data Sharing is designed to improve performance when many systems share a RACF database. RACF uses the Coupling Facility as a large Parallel Sysplex-wide store-through cache for the RACF database that reduces contention and I/O to DASD. When some commands are issued on one system, RACF communicates the command throughout the Parallel Sysplex. There is no need to issue the command on every system.

Figure 7-18 shows central Identity services for all z/OS subsystems running in all z/OS LPARs using a shared RACF database.

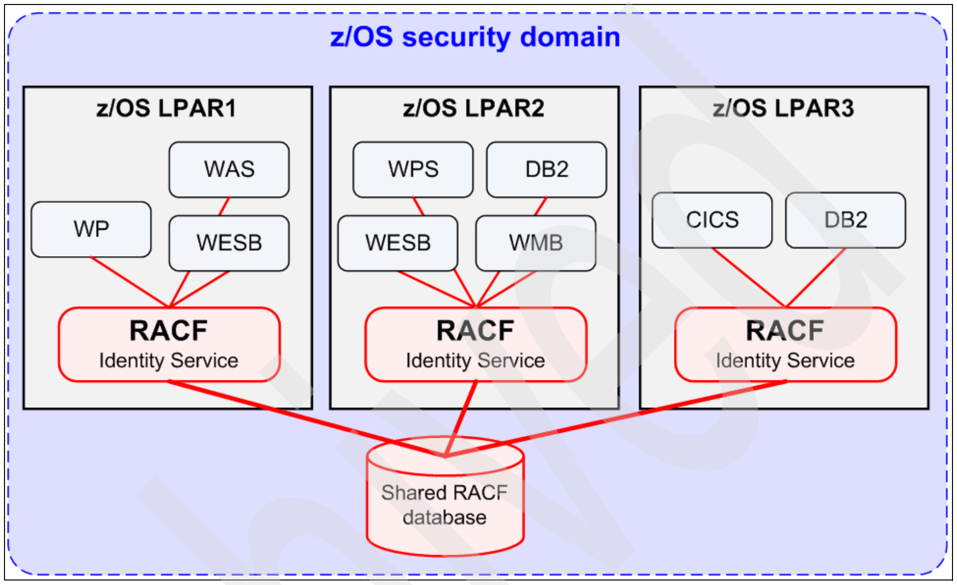


Figure 7-18 Central Identity services for all z/OS LPARs using a shared RACF database

RACF Remote Sharing Facility (RRSF) is another option to facilitate the use of RACF as a central Identity service.

RRSF allows you to administer and maintain RACF databases that are distributed throughout the enterprise. It provides improvements in system performance, system management, system availability, and usability. RRSF helps to ensure that data integrity is maintained across system or network failures and delays. The RRSF environment gives the ability to administer remote systems.

RRSF allows RACF to communicate via APPC with other z/OS systems that use RACF, allowing you to maintain remote RACF databases. RRSF extends the RACF operating environment beyond the traditional single host and shared DASD environments, to an environment made up of RRSF nodes that are capable of communicating with one another. This support provides administration of multiple RACF databases from anywhere in the RRSF network.

Benefits of RRSF support for the security administrator include:

- Administration from anywhere in the RRSF network.
  With RRSF, a security administrator logged on to one system in the RRSF network can direct allowed RACF TSO commands to remote RRSF nodes in the RRSF network. Administration of all the RACF systems in the RRSF network can take place from a single point of control.
- User ID associations.
  By supporting user ID associations and password synchronization, RRSF gives users with multiple user IDs the option of keeping their user ID passwords automatically synchronized across multiple systems.
- Automatic synchronization of databases.
  With automatic direction, RACF can keep databases synchronized automatically. When a command or application updates a database, RACF can automatically make the change to other databases.

Extend SAF-RACF z/OS Identity services with TDS

This solution software runs on z/OS only. This solution can include central Identity services for most z/OS and non-z/OS software products having an LDAP client for user registries.

RACF by itself is a central Identity service for the z/OS security domain itself. But RACF Identity service information can be made available to solutions outside of the z/OS environment through an LDAP interface to the RACF database. This is possible with *IBM Tivoli Directory Server (TDS)* for z/OS using an SDBM back end. TDS mainly provides Identity Directory services.

IBM Tivoli Directory Server for z/OS provides client access to an LDAP server. It consists of a new, rewritten LDAP server, an LDAP client, and utilities.

The z/OS Lightweight Directory Access Protocol (LDAP) server is based on a client/server model that provides client access to an LDAP server. An LDAP directory provides an easy way to maintain directory information in a central location for storage, update, retrieval, and exchange.

The LDAP server can be configured to provide read/write access to a RACF user, group, and connection profiles using the LDAP protocol. If the RACF data is shared across the sysplex, then users, groups, and connections in the sysplex can be managed using LDAP. The LDAP server's access to RACF is managed by an additional configurable back end called SDBM.

With TDS for z/OS configured with the SDBM back end, the RACF Identity service is extended and available to any product with an LDAP client. Many solutions running on distributed platforms or on Linux on System z can use LDAP as a user registry.

With TDS for z/OS and the SDBM back end, it becomes possible to authenticate to Linux on System z with a RACF user ID and password. It also becomes possible to authenticate to WebSphere Application Server running on AIX with a RACF user ID and password.

RACF along with TDS for z/OS can be a central Identity service for z/OS security domains, for Linux on System z security domains, and also distributed security domains.

Figure 7-19 shows how TDS for z/OS with the SDBM back end makes the RACF Identity service information available to solutions running in the Linux on System z security domain or in distributed security domains.

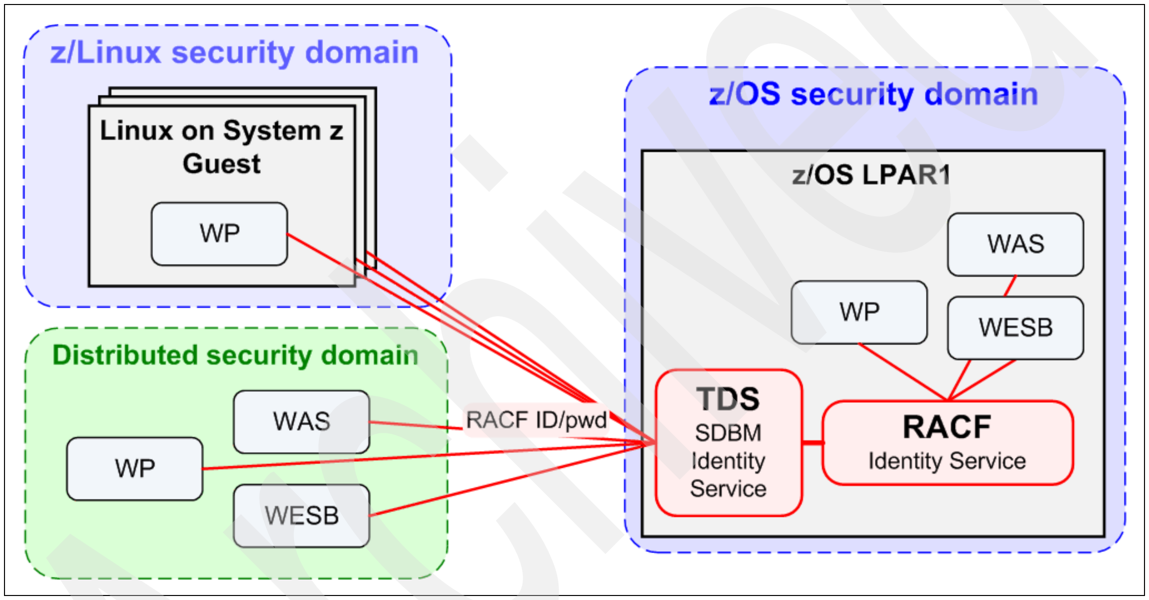


Figure 7-19 TDS for z/OS with a SDBM back end extending RACF Identity service to distributed platforms

The TDS for z/OS SDBM back end only allows access to RACF Identity service information. If there is a need for more flexibility with a customized LDAP schema, the TDS for z/OS TDBM back end can be used to store information in DB2 instead of RACF. With this configuration, there is still the possibility to store user identities and passwords in the RACF database using TDS for z/OS TDBM back end and native authentication.

Central Identity services with TDS on z/OS

The software for this solution runs on z/OS only and can include central Identity services for most z/OS and non-z/OS software products having an LDAP client for user registries.

Tivoli Directory Server for z/OS can be an LDAP standard compliant standalone central Identity service for some subsystems running on z/OS and for many solutions running on Linux on System z or distributed platforms. TDS mainly provides Identity Directory services.

There has been an LDAP directory implementation on System z for many years, dating back to the last versions of OS/390. With z/OS V1R8.0 and z/OS V1R9.0 there are two z/OS LDAP servers shipped:

- The *Integrated Security Services LDAP Server (or ISS LDAP)* is the heritage z/OS LDAP server, and is no longer being enhanced. It supports multiple back end datastores, such as RACF (called the SDBM) and DB2 (TDBM). This version of the directory is often referred to as just "z/OS LDAP."
- The *Tivoli Directory Server (TDS) for z/OS* is the newer z/OS LDAP server. It was introduced in March 2007, and new enhancements will go into this product going forward. It contains many of the features of ISS LDAP, but it provides an enhanced set of back end datastores and other functionality enhancements. One key change for this product over the older z/OS LDAP is that the disparate schemas used by the different back end datastores have been combined into a single server-wide schema.

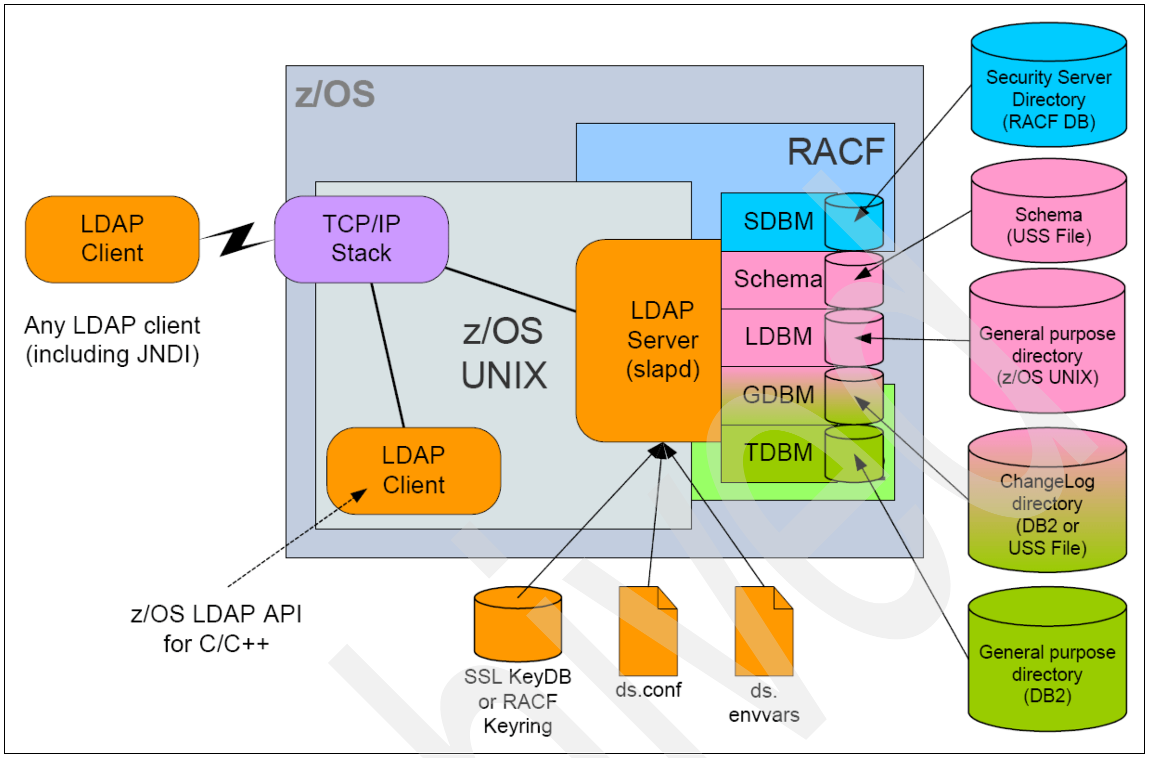
Tivoli Directory Server provides a central Identity service for rapid development and deployment of Web applications, security, and identity management on Linux on System z and distributed platforms. LDAP support ensures compatibility with industry standard LDAP-based applications.

The RAS enhancements in TDS for z/OS provide increased availability of the LDAP directory when a problem is detected in the LDAP server or in resources used by the LDAP server. The Parallel Sysplex support has been rewritten in the new TDS for z/OS server. It uses the cross-system Coupling Facility (XCF) to allow multiple LDAP servers to share directory data, including the LDAP server schema.

TDS for z/OS supports the following back ends:

- LDBM: General purpose file-based back end. This back end supports native authentication for the validation of RACF passwords.
- TDBM: General purpose DB2-based back end. This back end supports native authentication for the validation of RACF passwords.
- SDBM: RACF-based back end. This back end provides access to RACF Identity service information such as users and groups information.
- GDBM: Change log back end, DB2 or file-based back end.

Figure 7-20 on page 384 shows the components of TDS for z/OS.



z/OS
RACF
LDAP Client
Any LDAP client (including JNDI)
TCP/IP Stack
z/OS UNIX
LDAP Server (slapd)
SDBM
Schema
LDBM
GDBM
TDBM
LDAP Client
z/OS LDAP API for C/C++
SSL KeyDB or RACF Keyring
ds.conf
ds. envvars
Security Server Directory (RACF DB)
Schema (USS File)
General purpose directory (z/OS UNIX)
ChangeLog directory (DB2 or USS File)
General purpose directory (DB2)

Figure 7-20 Tivoli Directory Server for z/OS components

For high availability, scalability, and performance, TDS for z/OS leverages the Parallel Sysplex capabilities of its back ends. TDS for z/OS with the SDBM back end leverages RACF Data Sharing capabilities. TDS for z/OS with the TDBM back end leverages DB2 Data Sharing capabilities. Also, Sysplex Distributor (SD) can be used for load-balancing requests among multiple TDS for z/OS instances depending on performance goals.

Figure 7-21 on page 385 shows how TDS for z/OS leverages z/OS Parallel Sysplex capabilities in a high availability configuration.

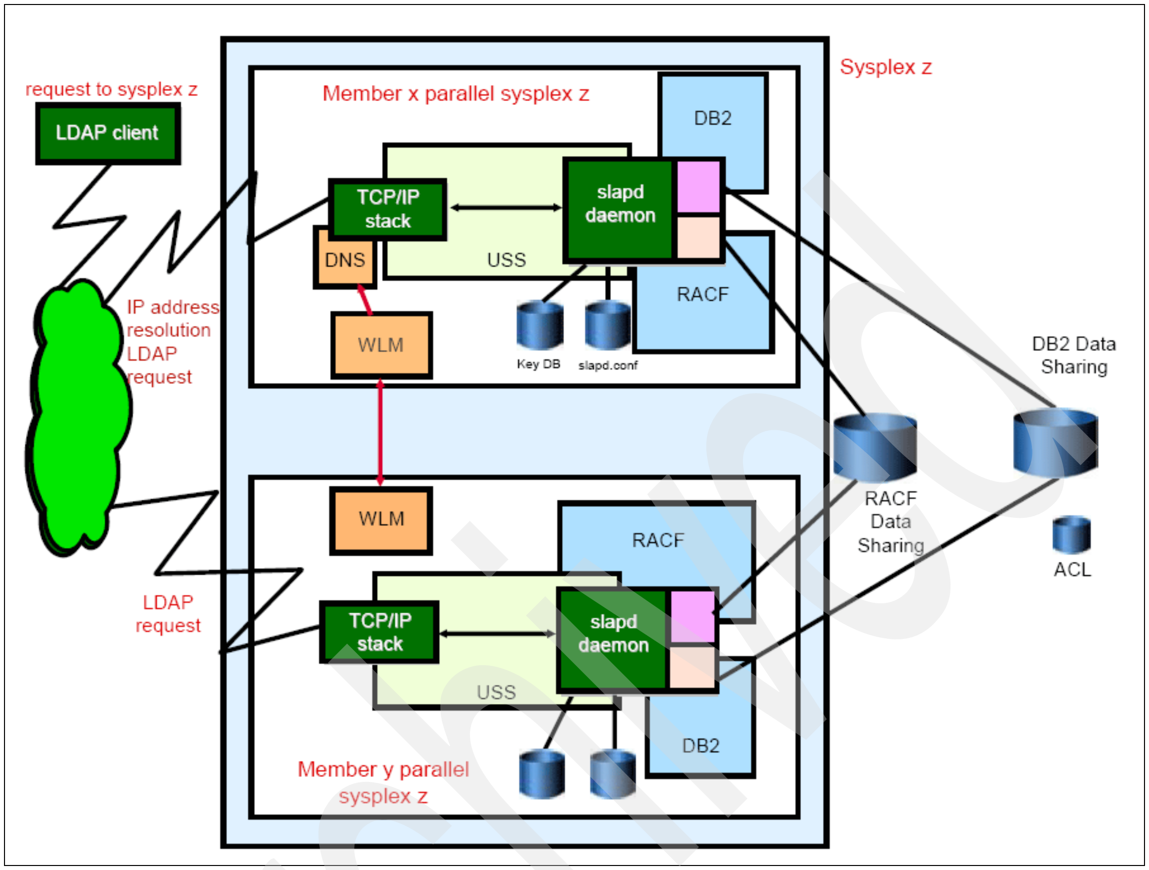


Figure 7-21 TDS for z/OS in a Parallel Sysplex configuration

A common use for TDS for z/OS is to provide a single centralized repository for user identities. If there are many Linux on System z images running, it makes sense to centralize the user accounts to simplify administration and have consistent auditing. In this model, the Linux on System z systems use the *Pluggable Authentication Modules (PAM)*, *Name Service Switch (NSS)*, and LDAP communications protocol to access user identity information back to the central TDS for z/OS server. Because standard LDAP is being used over TCP/IP, this framework could include machines outside of the System z (for example, other Linux and UNIX systems in the network).

Figure 7-22 on page 386 shows this centralized Identity service being used by the WebSphere Application Server running on Linux on System z, the Linux on System z operating system itself, and distributed client applications. As long as a solution has a local LDAP client, it can use the TDS for z/OS central Identity service.

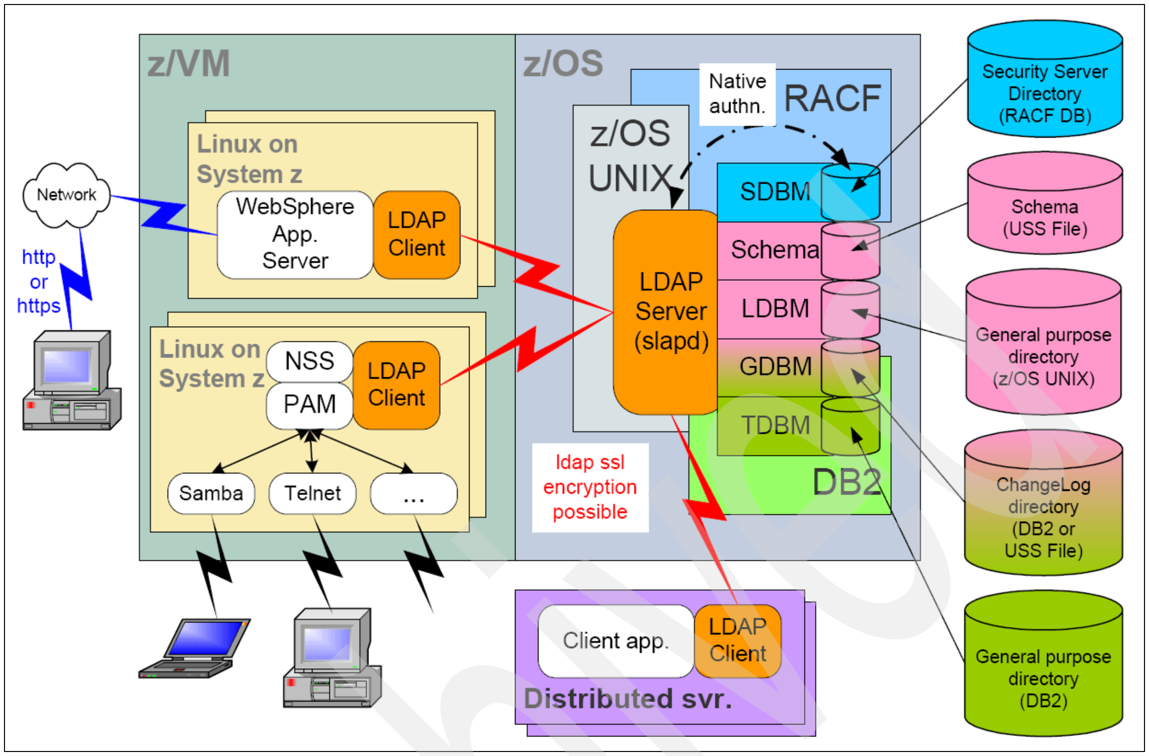


Figure 7-22 z/OS LDAP for centralized authentication

Central Identity services with TIM z/OS

The software for this solution runs on z/OS and can include central Identity services for most z/OS and non-z/OS Identity services products.

Tivoli Identity Manager (TIM) for z/OS is a full-featured central identity life cycle management solution that helps effectively manage user identities and their privileges across heterogeneous IT resources. It provides a secure, automated, and policy-based identity management and provisioning solution for z/OS, for Linux on System z, and distributed security domains. TIM provides complete identity management services.

Not only can TIM do identity provisioning to multiple Identity services such as RACF, but it can also do the access rules provisioning for identities. For example, when a new user is defined, TIM can provision the RACF identity as well as all the RACF attributes and profiles permissions that the new user has access to.

Here we consider an example scenario of how TIM facilitates identity management:

- Usually the scenario starts because an Identity service that has authoritative information about a user is triggered. For example, a new person is hired, someone leaves, a contractor or business partner is changed. TIM automatically monitors these types of changes and starts to adjust identities and access rights for that individual. Some accounts might be changed automatically and others might be requested manually.
- Suppose a user manually requests an identity and access for RACF (Identity change). The first thing that TIM does is take a look at whether or not that user is allowed to have access to RACF, which basically means “do the job responsibilities of this person require RACF?” If the user is allowed access to RACF, then the request is sent for approval. Maybe the user’s direct manager has to approve the request or maybe the business unit that owns the mainframe has to approve the request. Once the necessary approvals have been gathered, TIM will do everything necessary to create the RACF account.
- The important thing to remember is that the provisioning process does not stop with the creation or modification of the account. There is a need to be able to detect and correct changes made by a local administrator such as the local RACF administrator. If the RACF administrator grants the user additional privileges, or maybe removes some privileges, the user might have too much access, or not enough. TIM can detect the changes made by a local administrator. Once those changes have been detected, TIM can immediately take one of three actions: accept the changes, rollback the account to its proper state, or suspend the account for further review. Not all systems need the same levels of control. If it is a mission-critical system (such as accounting in most companies, product plans for manufacturers, patient records for health care providers, client financial systems for banks, and so forth), where espionage or breach could be fatal to the company, then it is necessary to ensure the systems are locked down tight and no one can subvert the process without knowing. But other systems are not so critical; maybe there is no need to track identities and access rights controls over e-mail or the development environments. In these cases, it might be desirable just to keep track of local changes so you have a clean, central audit trail.

Tivoli Identity Manager provides centralized role-based account provisioning for over seventy platforms. For many years Tivoli Identity Manager has been able to manage accounts on mainframes, and also Tivoli Identity Manager can run on z/OS and on Linux on System z. Tivoli Identity Manager for z/OS runs completely on z/OS. It uses z/OS LDAP, WebSphere Application Server for z/OS, and DB2 on z/OS. Functionally it is the same as the distributed product.

Figure 7-23 shows Tivoli Identity Manager for z/OS leveraging Parallel Sysplex, DB2 Data sharing, Sysplex Distributor, TDS for z/OS, and WebSphere Application Server for z/OS for high availability.

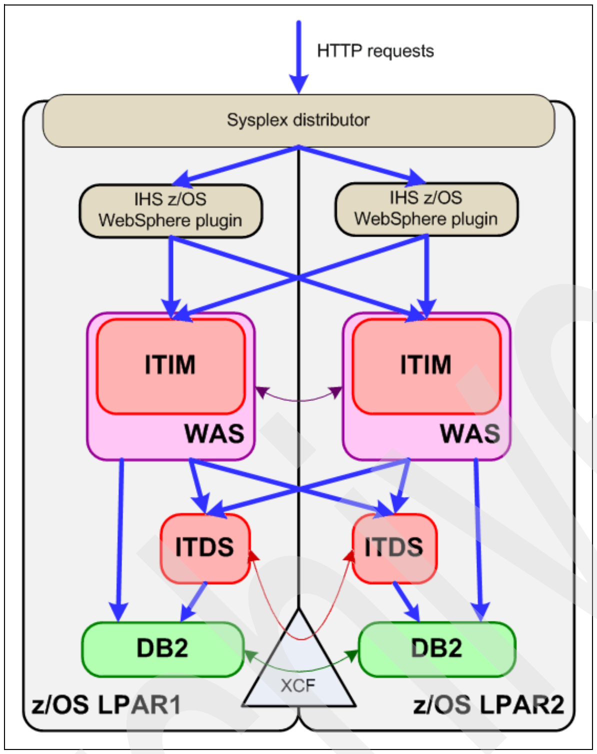


Figure 7-23 Tivoli Identity Manager for z/OS high availability configuration

When talking about managing System z security resources, it is common to think about accounts, passwords, attributes, and group memberships. These normally reside in an External Security Manager (ESM) such as RACF, CA's ACF2, or CA's TopSecret.

Tivoli Identity Manager provides an out-of-the-box adapter for RACF. This allows the provisioning of identities and permissions from the central Identity service (TIM) to the z/OS security domain Identity service (RACF). The adapter consists of two components: the DAML adapter that runs in z/OS UNIX System Services, and a command processor started task that runs in z/OS. Figure 7-24 on page 389 shows these components and where they reside.

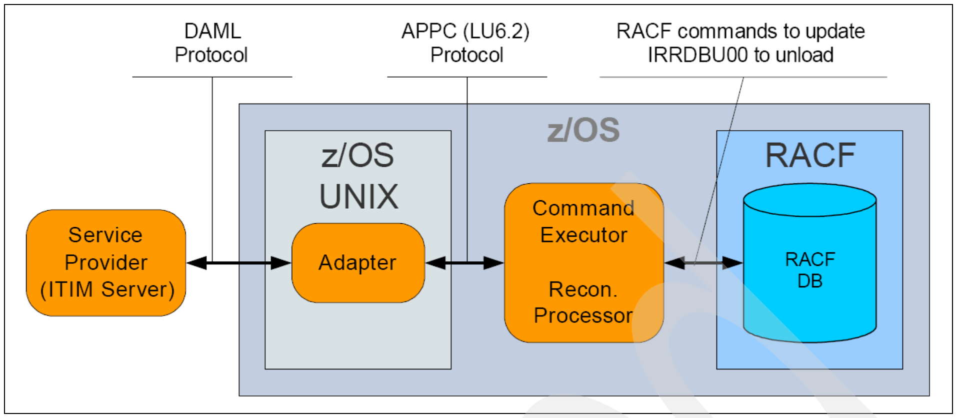


Figure 7-24 Tivoli Identity Manager RACF adapter components

The adapter also ships with a reverse password synchronization module from RACF to TIM that relies on the z/OS LDAP RACF changelog.

One of the key implementation concerns with Tivoli Identity Manager is the integration with an authoritative source for employee information, such as a Human Resource (HR) system. If the HR system is running on a mainframe, *IBM Tivoli Directory Integrator (ITDI)* could be used to build the HR feed. This depends on how the HR system can be accessed, but IBM Tivoli Directory Integrator provides a lot of flexibility, such as running on a distributed system (and using TCP/IP-based access protocols), running on Linux on System z, or running on z/OS. If none of the standard connectors can be used, the z/OS TSO/E Command Line Function Component could be used for command-line access to the system.

Linux on System z Identity services with TDS

The software for this solution runs on Linux on System z or on distributed platforms. This solution can include central Identity services for most z/OS and non-z/OS software products having an LDAP client for user registries.

Tivoli Directory Server (TDS) provides the LDAP identity infrastructure. It is built to serve as the identity data foundation for Web applications and identity management initiatives. TDS mainly provides Identity Directory services.

The Tivoli Directory Server for Linux on System z is based on the same implementation as the other Linux/UNIX implementations. It uses DB2 UDB as the back end data store.

It can be used as an identity and authentication source for the Linux on System z image (as with the Tivoli Directory Server on z/OS). It can also be used as a traditional directory by products such as Tivoli Identity Manager and Tivoli Access Manager for e-business or to provide other identity directory services.

Linux on System z Identity services with TIM

The software for this solution runs on Linux on System z or on distributed platforms. This solution can include central Identity services for most z/OS and non-z/OS Identity services products.

Tivoli Identity Manager (TIM) is a full-featured central identity life cycle management solution which helps effectively manage user identities and their privileges across heterogeneous IT resources. Tivoli Identity Manager running on Linux on System z is similar to Tivoli Identity Manager for z/OS from a functional perspective. From a non-functional perspective, Tivoli Identity Manager for z/OS leverages the high z/OS Quality of Service, whereas Tivoli Identity Manager on Linux on System z does not. TIM provides complete Identity Management services.

Tivoli Identity Manager supports running on Linux on System z. This includes the entire software stack: WebSphere Application Server, Tivoli Directory Server, and DB2. This is similar to other products running on Linux on System z. They cannot leverage the z/OS high Quality of Service. They allow customers to make use of the virtualization capabilities of the mainframe by hosting many Linux instances in one large server.

Identity services adapter with ITDI

The software for this solution runs on z/OS on Linux on System z or on distributed platforms and can be an identity adapter for most z/OS and non-z/OS identity services products.

IBM Tivoli Directory Integrator (ITDI) provides real-time synchronization between identity data sources so that enterprises can establish an authoritative, up-to-date, identity data infrastructure. ITDI mainly provides Identity Directory Integration services.

IBM Tivoli Directory Integrator can be configured as either a meta-directory or as a virtual directory. Today, firms need both capabilities. ITDI allows you to deploy either model, or even migrate between the two with one tool to learn. ITDI also provides many connections (by leveraging TIM agents) and reusable integration solutions with enhanced failover and more flexible business rules.

IBM Tivoli Directory Integrator's flexible architecture enables you to set up a system that can react and propagate directory changes in real time based on business logic. The *Assembly Line* executes the data integration flow based on the configuration of individual connectors, event handlers, parsers and the business logic driving the process. Applications can run ITDI synchronizations remotely and asynchronously. ITDI provides Web services support, including WS-provisioning and rich XML parsing. ITDI has failover capabilities for high availability and Assembly Line Pooling for bandwidth management.

ITDI can run in z/OS UNIX System Services or on Linux on System z.

The full list of connectors supported on z/OS can be found in the ITDI documentation, but the following connectors are of particular interest for z/OS:

- The z/OS TSO Command Line Function Component (z/OS TSO/E Command Line FC) is able to execute privileged z/OS TSO commands. This component addresses the need to manage RACF, TopSecret, and ACF2 users. This can be achieved by executing TSO commands. Architecturally, this FC consists of a Java layer, a UNIX System Services shared library, and a REXX script component. The Java layer passes the command to the shared library, the shared library passes it to the REXX script through APPC, and the REXX script executes the TSO/E command and passes back the result.
- The z/OS Changelog Connector is used to access the z/OS LDAP changelog for password synchronization and event notification.

The connectors supported with the Linux on System z implementation of IBM Tivoli Directory Integrator are the same as supported on the distributed IBM Tivoli Directory Integrator implementations and do not have z/OS-specific functionality.

Figure 7-25 shows how ITDI can be used for identity integration and synchronization from a central Identity service to Identity Directory services in AIX and z/OS security domains.

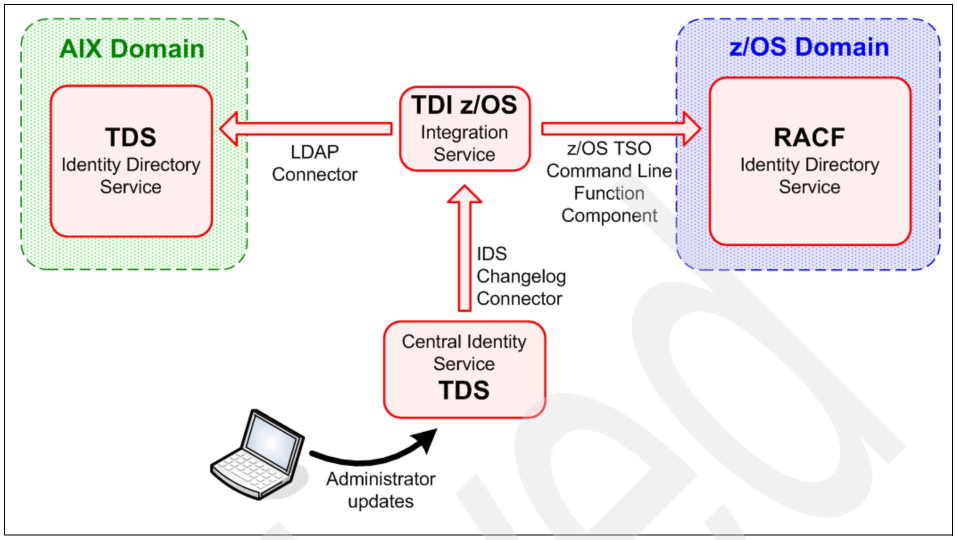


Figure 7-25 IBM Tivoli Directory Integrator along with AIX and z/OS security domains

The central Identity service can run on System z or not depending on non-functional requirements such as security level. It could be a TDS server and it could also be RACF.

ITDI can leverage RACF password enveloping in order to facilitate heterogeneous password synchronization. RACF password enveloping is also available to other applications, even z/OS-specific ones, using only RACF interfaces. RACF password enveloping is a function that allows authorized applications to recover a RACF user's password using an ldapsearch to the SDBM back end. It operates only with new passwords. The enveloped password is provided strongly encrypted, with digital signature, in a PKCS#7 envelope to authorized recipients. Eligibility for enveloping and authorization for retrieval of envelopes are controlled by RACF profiles.

RACF password enveloping used in conjunction with LDAP change log notification allows ITDI to synchronize z/OS RACF security domain with distributed security domain Identity Directory services.

Figure 7-26 on page 393 shows how ITDI works with RACF and z/OS LDAP SDBM to propagate the RACF password to other user registries.

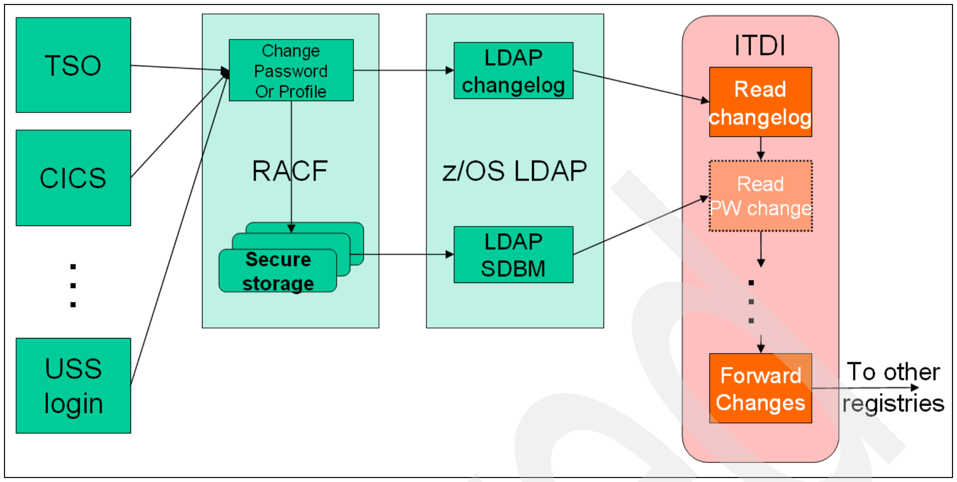


Figure 7-26 RACF profile and password synchronization scenario

Whenever there is a password creation or a password change in RACF, the password is stored in secure storage and a notification entry is created in the LDAP z/OS changelog. ITDI keeps reading the changelog entry and gets notified of the password change. ITDI then retrieves the RACF password envelop through the z/OS LDAP SDBM and forwards changes with the password envelop to other registries.

Figure 7-27 shows a real-world scenario where RACF on z/OS is the central Identity Directory service for the enterprise security architecture. RACF is of course also the Identity service for the z/OS security domain. The RACF central Identity Directory service is synchronized with the other security domains in the enterprise using ITDI.

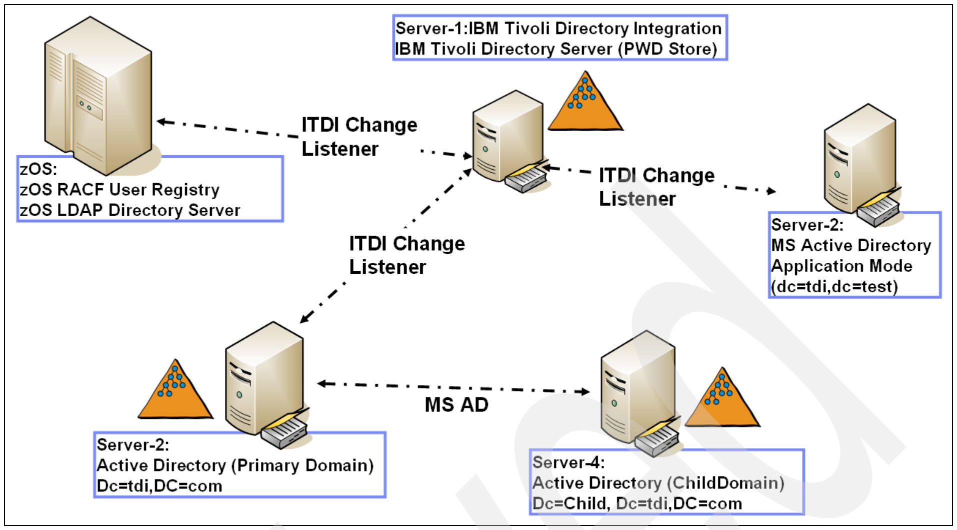


Figure 7-27 Real-world scenario: RACF, TDS and Active Directory synchronization using ITDI

First, the RACF central Identity Directory service is synchronized using ITDI with the security domain where TDS is the Identity Directory service. RACF password enveloping and z/OS LDAP SDBM are used to propagate the RACF passwords to TDS. Second, the TDS identity directory service is synchronized using ITDI with the security domain where Microsoft Active Directory is the Identity Directory service. Thus, accounts and passwords are synchronized from RACF to TDS to Active Directory.

7.3.5 Authentication services solutions

Authentication is the process of validating the identity claimed by the accessing entity. Authentication information is generally called *credentials*. It might contain the accessor's name and password, a *token* provided by a trusted party such as a Kerberos ticket, an x.509 certificate, or an LTPA token.

Authentication services have two main components:

- Authentication gateway
  This component intercepts requests and asks the user or the client for the proof of identity (credentials). Then the client information is sent for credential validation. Depending on the answer, the gateway certifies the client identity and assigns this identity to the request, or the gateway assigns an unauthenticated identity to the request. The authentication gateway can be embedded and internal in a solution such as an application server or it can be

an external standalone solution. If the authentication gateway is externalized, then it nicely fits in a DeMilitarized Zone (DMZ) for authenticating requests as soon as possible.

- Authentication provider
This component is a credentials validator and more. It receives credentials from the gateway and validates them against a user repository (check for passwords, for example) or validates them by verifying transformation results (decrypt a key, for example). It can also validate other attributes such as user revocation, date and time, and so on. Then the validator sends the positive or negative answer back to the gateway.

In this section we describe the solutions available to externalize and centralize these authentication components. We discuss the following solutions:

- Central external authentication provider
- Central external authentication gateway with TAM WebSeal
- Central external authentication gateway with WebSphere DataPower

Central external authentication provider

Most of the Identity Directory services provide authentication capabilities. They usually store secret information such as passwords and certificates associated with client identities for easier credentials validations.

Hence, most solutions described in "Identity services solutions" on page 378 apply also as authentication providers:

- Regarding the solution discussed in "Central Identity services with SAF-RACF z/OS" on page 379: SAF-RACF can be a central external authentication provider for most subsystems running on z/OS sysplex-wide.
- Regarding the solution discussed in "Extend SAF-RACF z/OS Identity services with TDS" on page 381: TDS for z/OS with an SDBM back end or with a TDBM back end and Native Authentication can be a central external authentication provider for LDAP-compliant authentication gateways running on z/OS, Linux on System z, or distributed platforms. SDBM back end or TDBM back end with Native Authentication allows verification of RACF passwords.
- Regarding the solution discussed in "Central Identity services with TDS on z/OS" on page 382: TDS for z/OS can be a central external authentication provider for LDAP-compliant authentication gateways running on z/OS, Linux on System z, or distributed platforms.
- Regarding the solution discussed in "Central Identity services with TIM z/OS" on page 386: TIM for z/OS relies on TDS for z/OS for Directory services. Therefore, TIM for z/OS can be a central external authentication provider for

LDAP-compliant authentication gateways running on z/OS, Linux on System z, or distributed platforms.

- Regarding the solution discussed in “Linux on System z Identity services with TDS” on page 389: TDS can be a central external authentication provider for LDAP-compliant authentication gateways running on z/OS, Linux on System z, or distributed platforms.
- Regarding the solution discussed in “Linux on System z Identity services with TIM” on page 390: TIM relies on TDS for Directory services. Therefore, TIM can be a central external authentication provider for LDAP-compliant authentication gateways running on z/OS, Linux on System z, or distributed platforms.

Central external authentication gateway with TAM WebSeal

The software for this solution runs on Linux on System z and on distributed platforms. This solution can be an external authentication gateway for most z/OS and non-z/OS Web servers, Web application servers, or Portal servers.

Tivoli Access Manager (TAM) WebSeal is a *Reverse Proxy Security Server (RPSS),* which can act as a central external authentication gateway.

With the use of *junctions*, TAM WebSeal can be the central external authentication gateway to many back end servers. TAM WebSeal is a multi-threaded Web proxy server that applies fine-grained security policy to the Tivoli Access Manager protected Web object space. WebSEAL can provide Single Sign-On (SSO) solutions and incorporate back-end Web application server resources into its security policy.

TAM WebSeal allows many login mechanisms such as HTTP Basic Authentication (BA), form-based login, client certificate login, and flexible login using the *External Authentication Interface (EAI)*. It handles external gateway authentication as well as some authorization at the URI level. It is also available as a plugin for several Web servers. It supports Single Sign-On for multiple Web applications.

TAM infrastructure supports the use of z/OS Security Server LDAP Server and Tivoli Directory Server for z/OS with a TDBM back end. Along with Native Authentication and identity synchronization, TAM WebSeal can provide external authentication with RACF identities and Single Sign-On or identity propagation to application servers such as WebSphere Application Server for z/OS.

Figure 7-28 on page 397 shows the TAM WebSeal architecture when integrating TAM WebSeal with WebSphere Application Server for z/OS and using RACF identities.

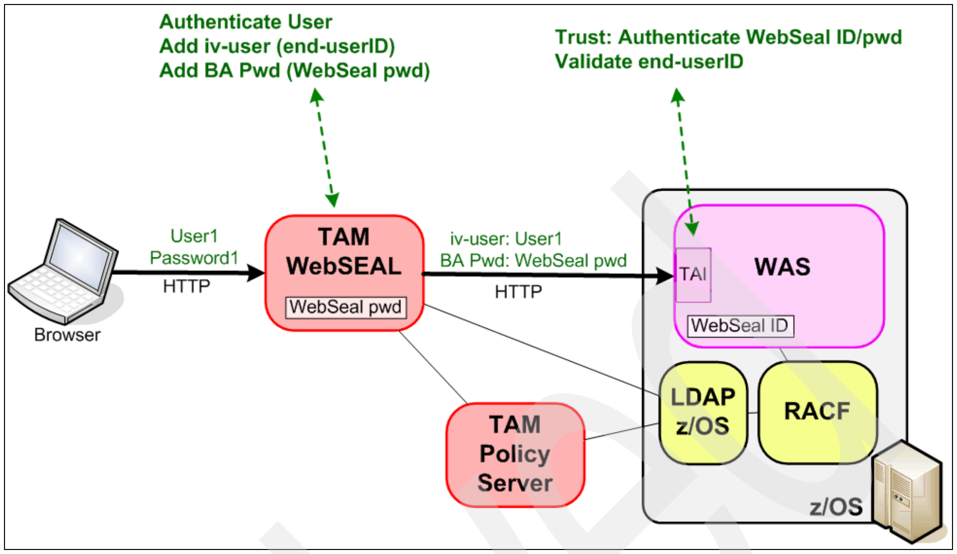


Figure 7-28 Central external authentication gateway with TAM WebSeal using RACF identities

When a request reaches the TAM WebSeal external authentication gateway, the end user is prompted for a RACF user ID and password. TAM WebSeal gets the credentials and sends them to the LDAP z/OS authentication provider for validation. LDAP z/OS is configured for TDBM Native Authentication and validates the password, comparing it with the password in the RACF database. Then the validation result is sent back to TAM WebSeal, which assigns the identity to the request or makes it unauthenticated. Then the request is forwarded to the back-end WebSphere Application Server for z/OS along with the RACF identity in the HTTP headers. WebSphere Application Server for z/OS is configured with RACF as the user registry (LocalOS) and RACF as the external authorization provider. The request is received by WebSphere Application Server for z/OS, which runs a *Trust Association Interceptor (TAI)* in order to retrieve the RACF identity from the header. Then authorization checks can happen in the WebSphere Application Server container with the RACF identity.

The advantage of this solution is that WebSphere Application Server for z/OS is configured with a Local Operating System user registry (RACF) and with external SAF Authorization (RACF). WebSphere Application Server for z/OS leverages the high security of RACF for managing J2EE roles and permissions in a central authorization provider. Another advantage of this solution is that RACF identities can be propagated to enterprise information systems such as CICS, IMS, or DB2 using native connectors.

TAM components can run on Linux on System z or distributed platforms.

This example architecture requires LDAP z/OS TDBM X.500 distinguished names to be synchronized with the RACF database. This synchronization is necessary in this example because the Trust Association Interceptor does a one-to-one mapping and extracts the RACF identity from the received HTTP header. If synchronization is not done, passwords can still be checked against RACF using Native Authentication.

IBM Tivoli Directory Integrator can be used to synchronize RACF with LDAP z/OS TDBM and vice-versa. With no synchronization, when a user is added to RACF, an LDAP directory entry is not automatically created. Or, when a user is added to TAM or LDAP, a RACF user is not automatically created. ITDI *Assembly Lines* can be used to enable such processing or Tivoli Identity Manager (TIM) might be used to enable coordinated user management across these multiple registries.

Figure 7-29 shows the ITDI Assembly Lines, which allow automatic synchronization when a user is added to RACF, when a user is added to LDAP, or when a user is added via the TAM console (pdadmin).

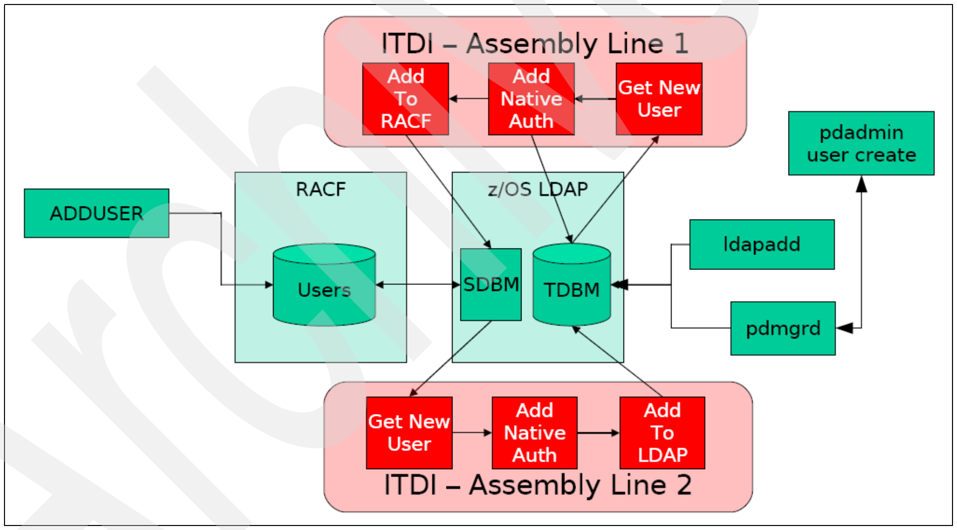


Figure 7-29 ITDI Assembly Lines to synchronize RACF and LDAP z/OS TDBM

Whenever there is an update from TAM or with ldapadd, Assembly Line 1 detects a change in the z/OS LDAP TDBM back end. It gets the new user identity and adds Native Authentication so that passwords are validated against RACF. Then it adds the user identity to RACF through the z/OS LDAP SDBM back end.

Whenever there is an update from z/OS with a RACF Adduser command, Assembly Line 2 detects a change in the z/OS LDAP SDBM back end. It gets the new user identity. It adds Native Authentication so that passwords are validated against RACF. Then it adds the user identity to the z/OS LDAP TDBM back end for TAM.

Information about using TAM also for authorization can be found in section “Central external authorization with TAM” on page 403.

Central external authentication gateway with WebSphere DataPower

This solution consists of a dedicated hardware appliance. This solution can be an external authentication gateway for most z/OS and non z/OS Web services or Web application providers.

WebSphere DataPower is a family of purpose-built and easy-to-deploy appliances that simplify, help secure, and accelerate XML and Web services deployments while extending the SOA infrastructure. WebSphere DataPower integration appliances models XI50 and XS40 can act as a central external authentication gateway.

WebSphere DataPower can be the central external authentication gateway to back-end servers or to an ESB deployed on System z or on distributed platforms. It provides a security enforcement point for XML and Web services transactions, including encryption, firewall filtering, digital signatures, schema validation, WS-Security, XML access control, and XPath. It offers robust service level management, policy management, and Web services management support, as well as detailed logging and auditing.

WebSphere DataPower provides access control functions that can be used to enable secure access to Web services-based applications to both internal and external clients. Both commercial and standards-based integration are supported, including LDAP, SAML, and WS-Security. Instead of URL-based or connection-level access control, fine-grained authorization allows DataPower to interrogate every individual SOAP/XML transaction and determine whether it should be allowed through based on payload contents, security policy, and identity information. For example, a purchase order that is (1) over $500, (2) digitally signed by the CFO's certificate, (3) targeted for vendor X, and (4) sent before 5 p.m. might be allowed through, while one immediately following it would be rejected.

Figure 7-30 on page 400 shows the WebSphere DataPower architecture when integrating with CICS or WebSphere Application Server for z/OS for Web Services external authentication.

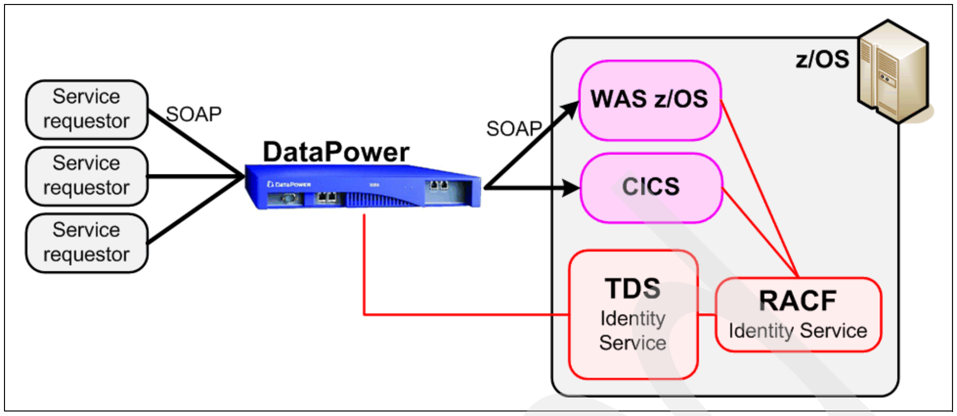


Figure 7-30 Central external authentication gateway with DataPower

WebSphere DataPower can also authenticate and authorize users against policies configured in Tivoli Access Manager.

7.3.6 Security authorization services solutions

Authorization is the process of checking whether an asserted (already authenticated) identity has access to a requested resource.

Authorization services have two main components:

- Authorization resource manager
  This component provides or prevents access to protected resources depending on decisions taken by the authorization provider. It intercepts executing requests and asks the authorization provider if a client identity can access a secured resource. If so, the authorization resource manager lets the request access the protected resource. If not, the resource manager rejects the access and stops the normal execution of the request. The resource manager is most of the time embedded with the resource it protects. A J2EE container is an authorization resource manager for J2EE applications protected with J2EE roles.
- Authorization provider
  This component makes authorization decisions requested by the authorization enforcer. The authorization enforcer provides the name of the protected resource, the identity trying to access the resource, and security context information. The authorization provider takes all this information into consideration, along with information from its own repository, and makes an appropriate decision to give access or to refuse access to the resource for the specified user. It sends the decision back to the authorization enforcer.

Again, authorization resource managers are usually close to the protected resource for enforcing security, so they are not centralized and externalized.

In this section we describe the solutions available to externalize and centralize the authorization providers. We discuss the following solutions:

- Central external authorization with SAF-RACF
- Central external authorization with TAM

Central external authorization with SAF-RACF

The software for this solution runs on z/OS only. This solution can provide central external authorization services for all z/OS subsystems supporting the SAF interface running on z/OS.

Resource Access Control Facility (RACF) can be a central external authorization provider service for most subsystems running on z/OS. RACF provides authorization services such as defining protected resources, permitting users and groups to access these resources, defining multiple levels of access control, and so on.

z/OS subsystems such as IMS, CICS, DB2, WebSphere MQ, WebSphere Application Server for z/OS, WebSphere Enterprise Service Bus for z/OS, WebSphere Process Server for z/OS, and many other z/OS products can use RACF as an authorization provider. Because all these products can store all their authorization definitions and rules in this common repository, RACF is an appropriate solution for handling central external authorization services across a z/OS infrastructure.

Not only is RACF able to be an external authorization provider service for all subsystems in one LPAR, but RACF can also be the external central authorization service for all subsystems running on any LPAR in a Parallel Sysplex.

For this purpose RACF can rely on a shared RACF database or Remote Sharing Facility. Additional information about these mechanisms is available in “Central Identity services with SAF-RACF z/OS” on page 379.

The *System Authorization Facility (SAF)* is part of the z/OS operating system that provides the interfaces to the callable services provided to perform authentication, authorization, and logging. The key element in SAF is the *SAF router*. The SAF router is always present, even when RACF is not present. The SAF router is a system service that provides a common focal point for all products providing resource control. This focal point encourages the use of common control functions shared across products and across systems. The resource managing components and subsystems call the router as part of decision-making functions, such as access control checking and authorization checking. These functions are called control points.

Figure 7-31 shows the z/OS Resource Manager, the SAF router and the RACF database.

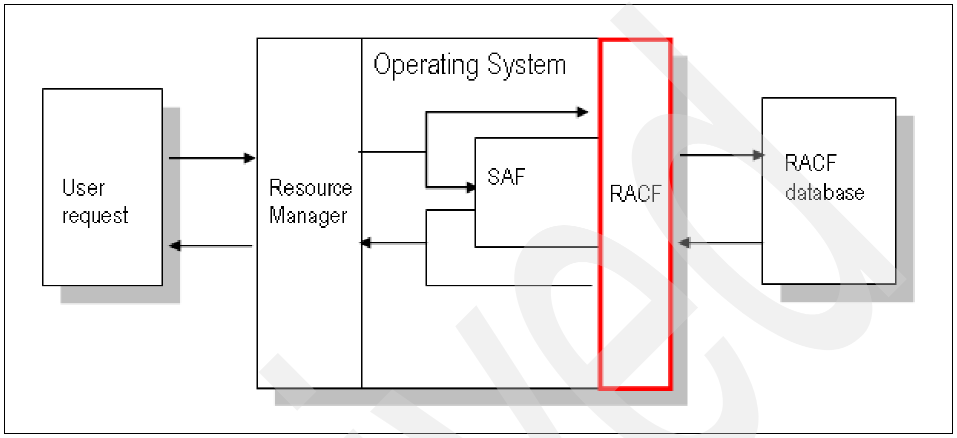


Figure 7-31 z/OS Resource Manager, SAF router and RACF database

SAF provides a central mechanism for customized security request processing. For example, with customer-written exit code, one could process requests in its own customized way. SAF made security management simpler because more applications used centralized security services rather than provide their own security mechanisms.

SAF relates to Resource Managers executing within z/OS and a security manager such as RACF. SAF is invoked by Resource Managers at system security control points, then SAF invokes the installation-selected security manager, for example RACF. The authorization decision is passed back to the Resource Manager by RACF through SAF.

A product using SAF, such as RACF, is an application that supports the RACROUTE interface and performs functions such as centralized auditing, resource authorization, and user identification and verification.

Using an SAF product also helps simplify and centralize your security. All the operator and command authorization syntax is created and managed within one product, rather than handled uniquely for each application.

Central external authorization with TAM

The software for this solution runs on Linux on System z and on distributed platforms. This solution can provide central external Authorization services for z/OS and non-z/OS software products supporting the JACC or JAAS API.

Tivoli Access Manager (TAM) can be a central external authorization provider service for some solutions running on z/OS, some running on Linux on System z, and some running on distributed platforms.

Tivoli Access Manager is a policy-based, access control security solution for a wide range of solutions. It delivers unified authentication and authorization access to diverse applications within the entire enterprise. *Tivoli Access Manager for Business Integration* protects access to WebSphere MQ resources. *Tivoli Access Manager for Operating Systems* protects access to Linux and UNIX operating system resources. *Tivoli Access Manager for e-business* protects access to Web application resources. So, many solutions can benefit from central external authorization with Tivoli Access Manager.

Figure 7-32 shows Tivoli Access Manager being the central external authorization provider for several solutions, including TAM WebSeal URI authorizations, WebSphere Application Server for z/OS resources, WebSphere MQ, and some other operating systems.

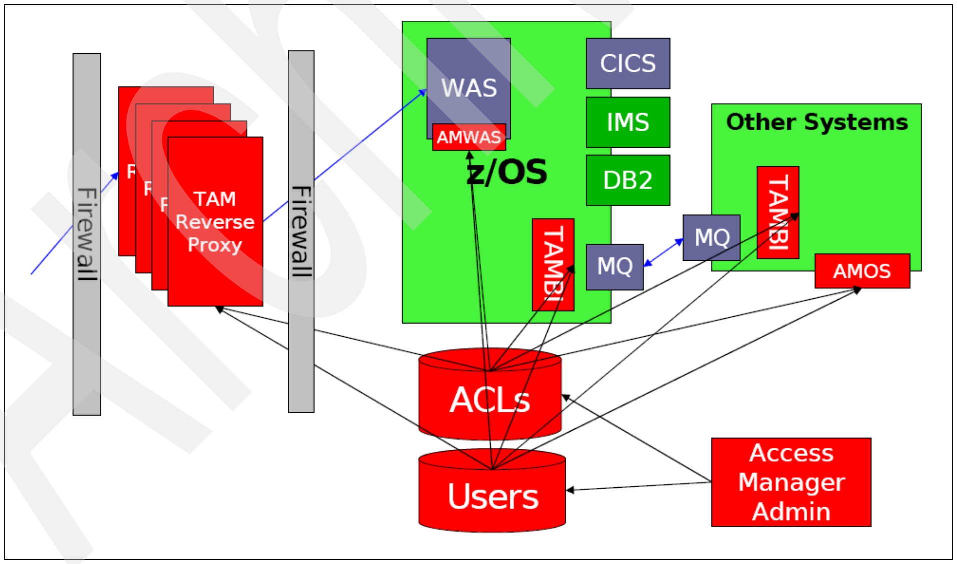


Figure 7-32 Tivoli Access Manager as central external authorization provider

More specifically for WebSphere Application Server running on z/OS or Linux on System z, TAM can use mechanisms such a the Trust Association Interceptor

(TAI), the Java Authorization Contract for Containers (JACC) and the Java Authentication and Authorization Service (JAAS).

The TAI is used during authentication time, which is not of interest in this section.

The *Java Authorization Contract for Containers (JACC)* specification enables third-party security providers to handle the J2EE authorization. The JACC specification requires that the containers in both the application server and the provider satisfy some requirements. Specifically, the containers are required to propagate the security policy information to the provider during the application deployment and to call the provider for all authorization decisions. The providers are required to store the policy information in their repository during application deployment. The providers then use this information to make authorization decisions when called by the container. Tivoli Access Manager can be a central external JACC authorization provider for WebSphere Application Server.

For applications developed in house, Tivoli Access Manager also provides APIs that grant access to its services. TAM supports the J2EE standard JAAS to allow native Java applications to access TAM for authorization decisions.

Figure 7-33 shows Tivoli Access Manager integrated with WebSphere Application Server for z/OS for authorization with JACC and JAAS. Integration capabilities for authorization are the same with WebSphere Application Server running on Linux on System z.

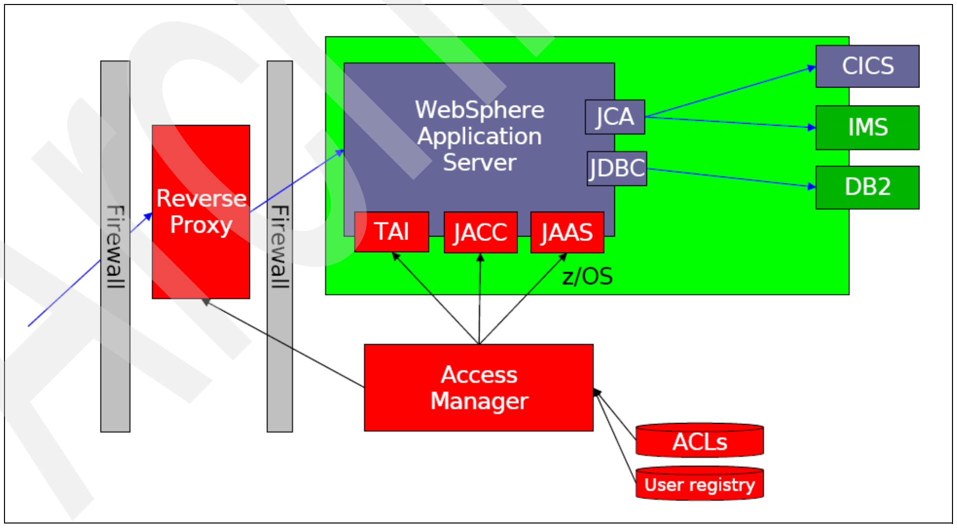


Figure 7-33 Tivoli Access Manager integrated with WebSphere Application Server on z/OS

Tivoli Access Manager components such as the Authorization server, the Policy server, and the WebSeal authentication proxy do not run on z/OS, but they can run on Linux on System z. This allows hosting of a TAM-protected environment entirely on System z systems in order to better leverage the System z Quality of Service characteristics. Hipersockets support can be used between Linux on System z guests and z/OS for higher security and higher performance. z/VM virtual network connections enable precise routing between Webseal-dedicated guests and other Linux guests. Linux firewalls running on System z avoid network communications out of the System z machine for network level security.

Figure 7-34 shows Tivoli Access Manager deployed on System z with LDAP on z/OS being used along with Native Authentication for password validation in RACF.

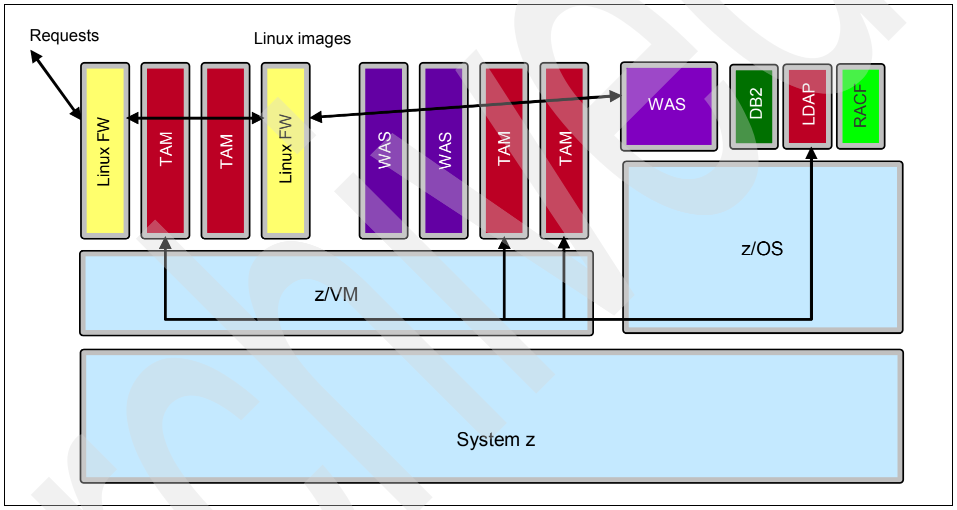


Figure 7-34 Tivoli Access Manager on System z

Information about using TAM for external authentication can be found in “Central external authentication gateway with TAM WebSeal” on page 396.

7.3.7 Security federation solutions

SOA connects loosely coupled services to construct new applications. These services have their own user registries that are often administered in isolation from those of other services in the SOA environment. Users and service entities in a heterogeneous environment or across multiple security domains are likely to have different identities in the various services that make up a composite application.

Establishing the identity of the service requester in each service request is a fundamental step in ensuring that business requirements such as authorization, audit, and compliance can be implemented.

Identity Federation services are required in the SOA infrastructure so that services can be easily interconnected with the correct identities being propagated. Within federated identity management the focus is on the user identity.

Two sets of functions are needed for Identity Federation services:

- Federated Single Sign-On (SSO), also called identity propagation, which takes care of security mediations for flowing identities between components with different identity requirements
- Federated identity provisioning, which takes care of provisioning identities between identity providers and service providers

Identity federation is particularly relevant for secured conversations between services in the z/OS security domain and services in other security domains.

Federated Single Sign-On or identity propagation

The software for this solution runs on Linux on System z or on distributed platforms, and partly on z/OS. This solution provides central Federation services for most z/OS and non-z/OS Identity services products.

Multiple security domains such as z/OS security domains, Linux on System z security domains, and distributed security domains make Single Sign-On or identity propagation a challenge in an SOA infrastructure. Also, many organizations are moving from using application-level user IDs and passwords to using individual user identities, to satisfy new compliance standards. Identity federation can allow users from one federation partner to seamlessly access resources from another partner in a secure and trustworthy manner. A solution for SOA identity propagation must be:

- Capable of understanding and operating with a variety of formats for representing identity
- Capable of translating between different identities
- Based on SOA principles itself to deliver a flexible, infrastructure-based solution de-coupled from application business logic
- Constructed using open standards to provide maximum inter operability with the platforms and systems on which SOA solutions are constructed

The IBM SOA identity propagation solution is built on open standards. The *WS-Trust standard* (part of the WS-Security family of standards) is the open mechanism by which security tokens can be validated, issued, and renewed, and also trust relationships can be established, assessed, and brokered. WS-Trust is defined by a Web services interface. The service that implements the WS-Trust interface is known as a *Security Token Service (STS)*.

In the IBM SOA identity propagation solution, the STS is a component of the *IBM Tivoli Federated Identity Manager (TFIM)* product. Tivoli Federated Identity Manager is the IBM implementation of the federated trust communications between various parties using industry standards such as SAML, Liberty, WS-Federation, and WS-Trust. Tivoli Federated Identity Manager is a standards-based, access control solution for federated Single Sign-On) and trust management in a Web services and SOA environment.

Tivoli Federated Identity Manager is supported on Linux on System z. As with the other products, this implementation is effectively the same as on distributed systems. Tivoli Federated Identity Manager is also supported on z/OS, but with limited functionality. The run time and management service component, the Web Services Security Management (WSSM) component, and the management console are supported on z/OS. The provisioning components, federated Single Sign-On, and the Common Auditing and Reporting Service are not supported on z/OS.

Integration with the SOA identity propagation solution follows one of three general patterns:

- Service requester
  It recognizes the need for consumers of a service to send a service what it expects. This pattern represents the authenticated identity in the service component in an identity token and uses an STS to transform the authenticated identity to an identity token suitable for sending in the service request.
- Service provider
  An incoming identity token is sent to the STS for validation and mapping to a local identity. This pattern is used in cases where the receiving service component is expected to accept an identity token that it is not able to natively support.
- Intermediary
  Enables identity propagation through an intermediary, such as an Enterprise Service Bus. This pattern is a combination of the service requester and service provider patterns, where incoming identity tokens are required to be validated (as in the service provider pattern) before the identity token for an outgoing request is generated (as in the service requester pattern). ESBs are required to perform identity mediation because they often sit at the boundary of different administrative domains.

TFIM offers identity mapping and role-based mapping options to allow a wide variety of federation models to be implemented. In some cases, an organization can achieve dramatic reductions in user administration, complexity, and costs, by adopting federated single sign-on. To aid in compliance with regulatory requirements and corporate governance standards, TFIM also provides an integrated audit data collection and reporting component.

The TFIM STS provides identity mediation services for an SOA. TFIM also includes STS consumer modules to allow the STS functionality to be leveraged from WebSphere Application Server. For example, these STS consumer modules allow SAML assertions in WS-Security headers of Web services messages to be consumed or generated by WebSphere Application Server. The functionality provided by the STS can also be accessed from XML firewalls or gateways, including IBM DataPower SOA appliances, to provide identity mediation services to these boundary devices for XML-based interactions with external organizations. The STS can also be invoked directly from applications or other middleware via the protocol defined by the WS-Trust standard.

The Tivoli Federated Identity Manager Security Token Service can also be used to map distributed user IDs to z/OS Security Server RACF user IDs and associated RACF PassTickets (one-time passwords for authentication to RACF). The RACF ID and PassTicket can then be used to connect to z/OS hosted resources using individual user identities. The TFIM STS in this use case can be hosted on z/OS or a supported distributed platform. z/OS support includes RACF PassTicket access to CICS and IMS transactions. The STS performs validation of security tokens, transformation of format, and mapping of identity.

Figure 7-35 on page 409 shows federated single sign-on or identity propagation using a TFIM STS from WebSphere Portal to CICS.

In this scenario, the user authenticates to the company Portal using a user identifier (for example, jsmith) and a password. In order to access the exposed Web service (for example, a CICS application), the Portal needs to transform the identity to a valid representation for the receiving system (for example, a RACF passticket with the user identifier joey1234). This exchange is done using a security token service.

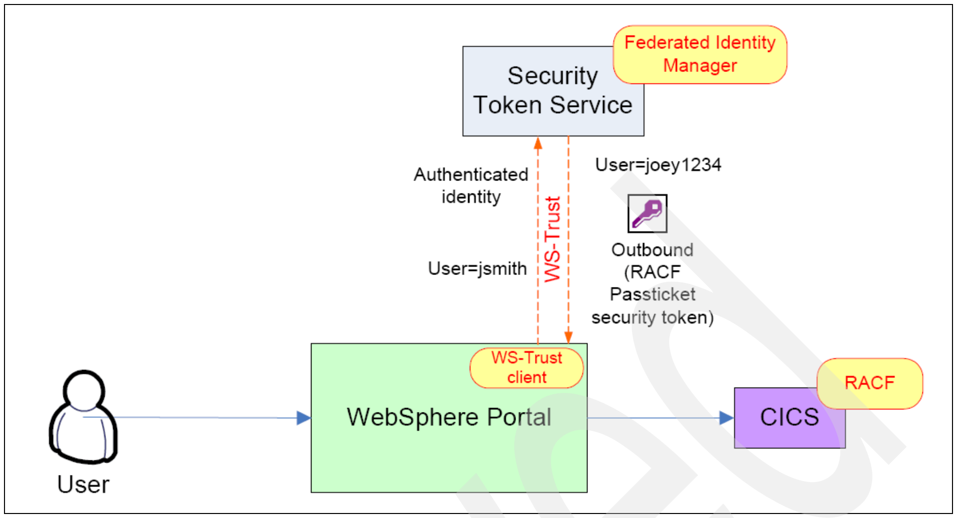
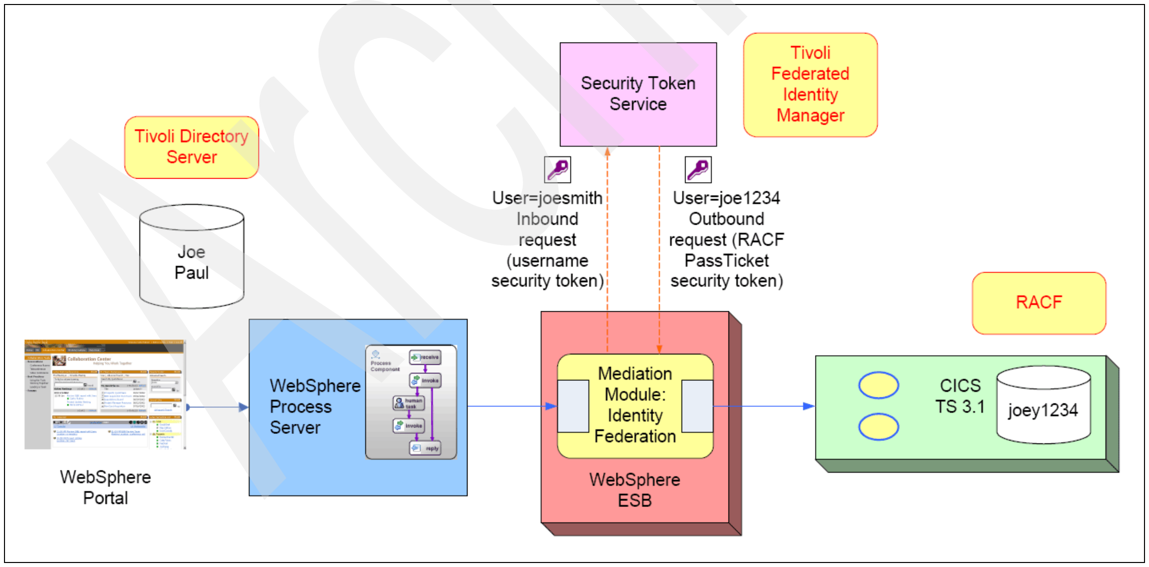


Figure 7-35 Federated Single Sign-On from WP to CICS with TFIM STS

In Figure 7-36, the request from the internal consumer carries an identity that is used by the WebSphere Process Server. When WebSphere Process Server needs to make service requests to the CICS system, an identity mapping is required. This is translating the user's Tivoli Directory Server identity into one suitable for RACF.



Security Token Service
Tivoli Federated Identity Manager
Tivoli Directory Server
Joe
Paul
User=joesmith Inbound request (username security token)
User=joe1234 Outbound request (RACF PassTicket security token)
RACF
WebSphere Portal
WebSphere Process Server
Mediation Module: Identity Federation
WebSphere ESB
CICS TS 3.1
joey1234

Figure 7-36 Federated Single Sign-On from WESB to CICS with TFIM STS

The correct architectural pattern to achieve identity mediation with WebSphere Process Server is to leverage the WebSphere ESB mediation. The WebSphere ESB connects to the Security Token Service, which performs the identity mapping and token transformation.

Figure 7-37 shows the general architecture of a federation solution for a z/OS security domain with an AIX security domain. Tivoli Identity Manager (TIM) handles the provisioning of identities from a central Identity Directory service to the z/OS security domain Identity service (RACF) and to the AIX security domain Identity service (TDS). TIM also handles the provisioning of proof of identities (such as passwords) for the authentication services of both security domains. TIM also does the provisioning of access rules or permissions to the z/OS security domain authorization service (RACF) and to the AIX security domain authorization service (TAM). TIM uses adapters for the provisioning.

When components from the AIX security domain need to send a request (Web service request, for example) to components in the z/OS security domain, the TFIM STS component transforms the identity and its format understandable in the AIX security domain (such as a X.500 distinguished name) to an identity and format understandable in the z/OS security domain (such as a RACF ID).

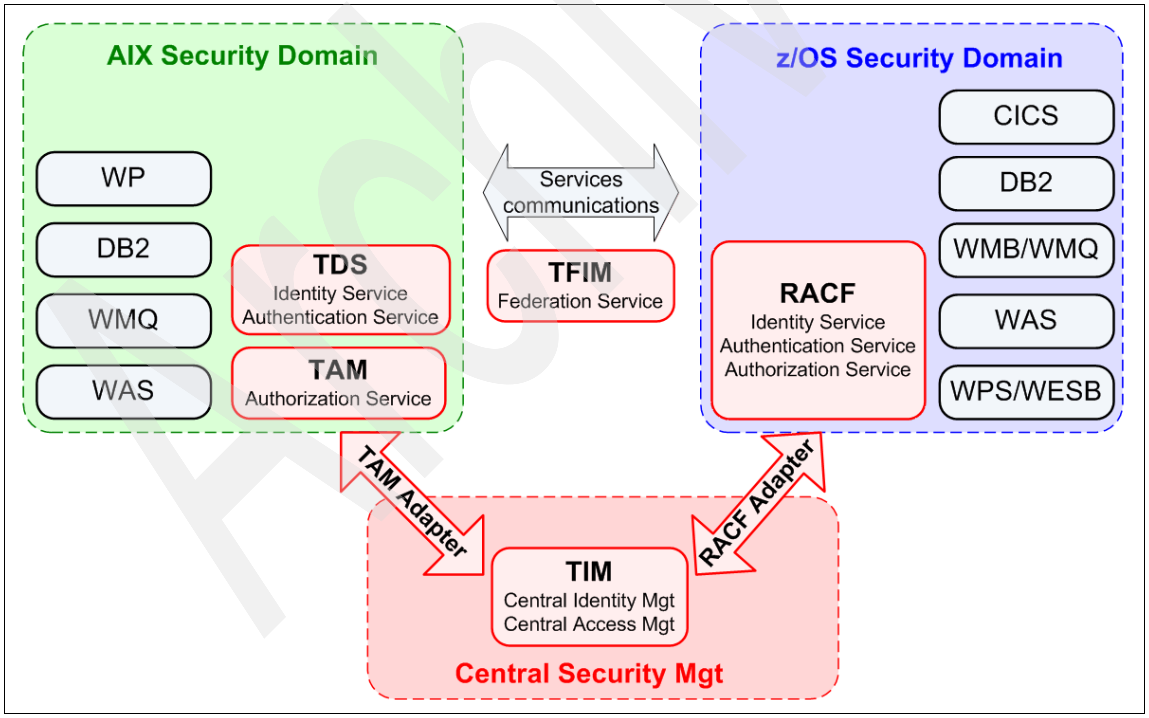


Figure 7-37 Federation of heterogeneous security domains with TFIM

Federated identity provisioning

The software for this solution runs on Linux on System z or on distributed platforms, and partly on z/OS. This solution can provide central Federation services for most z/OS and non-z/OS Identity services products.

Federated identity provisioning is about remotely having the capability of managing attributes of, for example, a user as part of an identity management process. Federated identity provisioning extends these provisioning management activities beyond a trusted security domain.

Federated identity provisioning makes it possible to extend local account provisioning at an identity provider to include federated account provisioning out to multiple service provider partners. A service provider, when notified of the federated provisioning request, can perform the local provisioning necessary to supply its service to the specified employee.

Provisioning requests sent between identity providers and service providers must be secure and be based on open standards. A standard that satisfies these requirements is *WS-Provisioning*. WS-Provisioning includes operations for adding, modifying, deleting, and querying provisioning data. It also specifies a notification interface for subscribing to provisioning events. Provisioning data is described using XML and other types of schema. This facilitates the translation of data between different provisioning systems. Figure 7-38 provides an overview of the WS-Provisioning support provided in Tivoli Federated Identity Manager (TFIM).

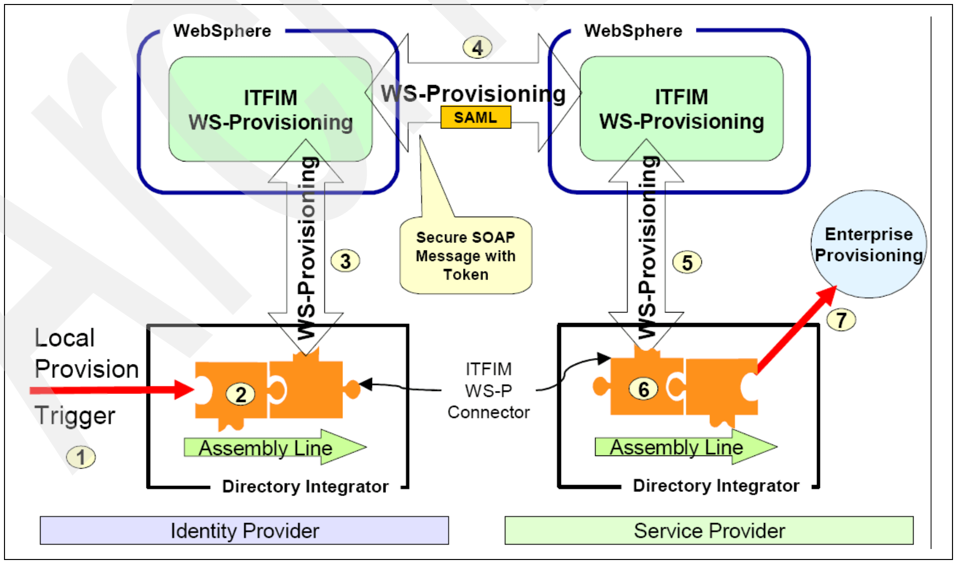


Figure 7-38 TFIM Federated provisioning

The Tivoli Federated Identity Manager components are the Tivoli Federated Identity Manager WS-Provisioning Web service that runs on WebSphere Application Server and the Tivoli Federated Identity Manager WS-Provisioning Connector that runs on Tivoli Directory Integrator. Both of these provide a full implementation of the three interfaces defined by the WS-Provisioning standard.

7.3.8 Security integrity and confidentiality solutions

Integrity ensures that transmitted or stored information has not been altered in an unauthorized or accidental manner. *Confidentiality* refers to the concept that an unauthorized party cannot obtain the meaning of the transferred or stored data.

Integrity and confidentiality are commonly obtained with the use of cryptography mechanisms involving encryption and keys. This section describes the following solutions that facilitate the use of such mechanisms:

- Central key management services with SAF-RACF
- Central PKI services with z/OS
- Central encryption services with the Encryption Facility
- Central database encryption services
- Central encryption to tape services with TS1120 and EKM

Central key management services with SAF-RACF

The software for this solution runs on z/OS. This solution provides central key management and key serving services for most z/OS subsystems supporting the SAF interface. This solution also provides central key management services for some non-z/OS products.

RACF can be used to create, register, store, and administer digital certificates and their associated private keys, and build certificate requests that can be sent to a certificate authority for signing. These digital certificates can be stored in RACF for use by z/OS subsystems or they can be exported for use by Linux on System z or distributed products. RACF allows users to act as their own Certificate Authority (CA) for z/OS or non-z/OS products. Thus users do not have to buy digital certificates or similar services from other sources or run CAs on other platforms.

RACF can also be used to manage keyrings of stored digital certificates. Digital certificates and key rings are managed in RACF primarily by using the RACDCERT command or by using an application that invokes the *R_datalib callable service* or the *initACEE callable service*. The R_datalib callable service provides an API to the *CDSA (Common Data Security Architecture)* data library functions, and is used by Secure Sockets Layer (SSL) and System SSL to establish secure sessions between servers. The initACEE callable service can be used to manage digital certificates for RACF-authenticated users.

RACF has three categories for managing digital certificates:

- **User certificate**
  A certificate that is associated with a RACF user ID and is used to authenticate the user's identity. The RACF user ID can represent a traditional user or be assigned to a server or started procedure.
- **Certificate-Authority certificate**
  A certificate that is associated with a certificate authority and is used to verify signatures in other certificates.
- **Site certificate**
  A certificate that is associated with an off-platform server or other network entity, such as a peer VPN server. This category of certificate can also be used to share a single certificate and its private key among multiple RACF user IDs. When used for sharing, a certificate might be referred to as a placeholder certificate.

The RACDCERT (RACF Digital Certificate) command is used to store and maintain digital certificate information in RACF, and should be used for all maintenance of certificate profiles and related USER profile fields. This command can be used to list, add, check, alter, delete, generate, write, bind, import, and export digital certificates.

For using RACF-generated digital certificates on Linux on System z or distributed platforms, certificates can be exported and then transferred in three formats. The CERT keywords indicate that only a certificate is to be exported. The PKCS #7 keywords indicate to export a certificate and its CA chain. PKCS #7 processing attempts to package any Certificate-Authority certificate necessary to complete the basing chain to the exported certificate. The PKCS #12 keywords indicate to export the certificate and the private key (which must exist and must not be an ICSF or PCICC key). The package produced by specifying one of the PKCS #12 keywords is encrypted using the password specified according to the PKCS #12 standard. PKCS #12 processing attempts to package any Certificate-Authority certificate necessary to complete the basing chain to the exported certificate.

RACF can be used along with *Integrated Cryptographic Service Facility (ICSF)* to leverage the hardware cryptographic features to provide secure, high-speed cryptographic services in the z/OS environment. ICSF provides the APIs by which applications request the cryptographic services. ICSF is also the means by which the secure cryptographic features are loaded with master key values, allowing the hardware features to be used by applications. The cryptographic feature is secure, high-speed hardware that performs the actual cryptographic functions. The processor hardware determines the cryptographic feature available to the applications.

Central PKI services with z/OS

The software for this solution runs on z/OS. This solution provides central Public Key Infrastructure (PKI) services for z/OS and non-z/OS products.

A PKI enables users of a public or a private network to securely and privately exchange data through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority.

PKI services support the life cycle management of large numbers of digital certificates. Digital certificates which are based on Public Key Encryption technology provide a foundation for secure and scalable user identification and authentication and secure and verifiable data exchange.

PKI services is a technology that allows users to act as their own Certificate Authority (CA) for their internal and external users, issuing and administering digital certificates in accordance with their organizational policies. Thus users do not have to buy digital certificates or similar services from other sources or run CAs on other platforms. z/OS PKI services is certified by Identrus as an Identrus-compliant CA.

PKI services is included in z/OS Security Server at no additional cost. PKI services for z/OS combines PKI encryption technology with the z/OS Quality of Service, including availability and scalability, thus providing the following capabilities:

- It allows you to establish a PKI infrastructure and serve as a Certificate Authority (CA) for users, issuing and administering digital certificates.
- It provides digital certificates life cycle management.
- It supports RACF for administering system and resource access.
- It is scalable to drive thousands of certificates and is secured with System z cryptography.
- It uses the *Online Certificate Status Protocol (OCSP)* for dynamic checking of certificate status (revocation).
- It can use the 4758 Cryptographic co-processor for private keys.
- End-users requests are driven via customizable Web pages and there can be an automatic or administrator approval process.

Figure 7-39 on page 415 shows the components of the central PKI services on z/OS.

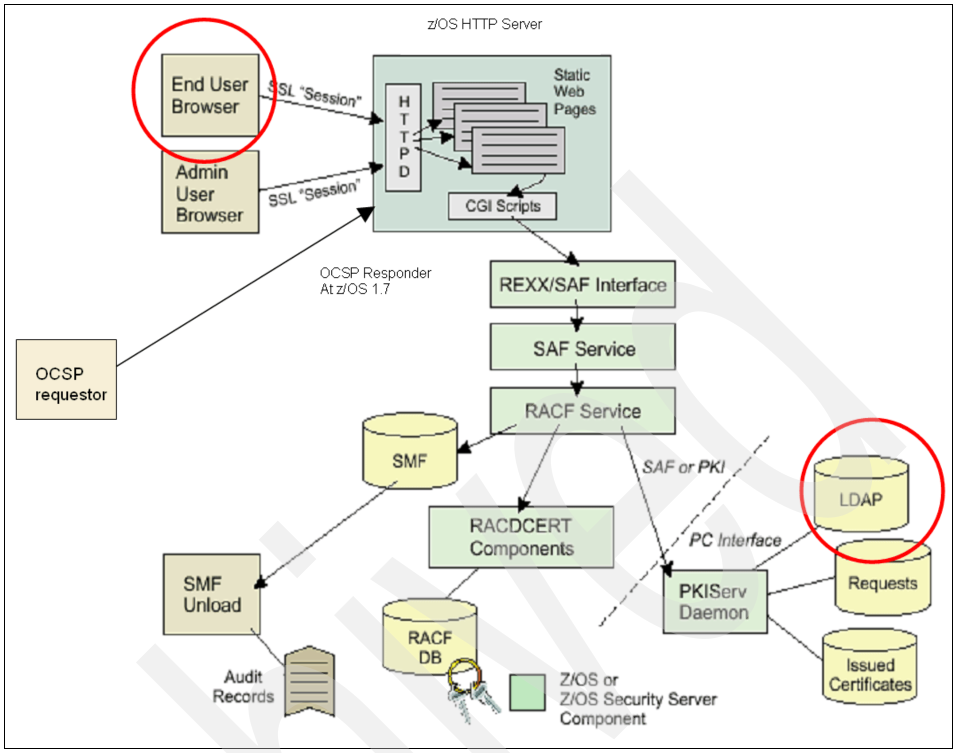


Figure 7-39 Central Public Key Infrastructure services on z/OS

Users request and receive certificates from a Web browser (HTTP). Users can request certificate revocation or certificate suspension. The *Certificate Revocation List (CRL)* is published in a LDAP directory.

Central encryption services with the Encryption Facility

The software for this solution runs on z/OS and the client runs on z/OS, on Linux on System z, or on distributed platforms. This solution provides central data encryption services for most z/OS needs and some non-z/OS needs.

The *Encryption Facility for z/OS* is a host-based software solution designed to encrypt sensitive data to disk, to tape, or to removable media. It can be used for the z/OS platform only or for exchanges between the z/OS platform and distributed platforms.

Encryption Facility for z/OS consists of two independent features:

- The Encryption Services feature supports encrypting and decrypting certain file formats on z/OS. This allows transfer of data to remote sites and eventually to partners and vendors, and then archive. The Encryption

Services feature supports both the System z format and the OpenPGP format. The System z format supports hardware-accelerated compression before encryption.

- The DFSMSdss Encryption feature enables the encryption of DFSMSdss dump data sets. This feature supports hardware-accelerated compression before encryption to tape.

Also available is the *Encryption Facility for z/OS Client*. The Encryption Facility for z/OS Client is a no-cost, separately licensed program (which is offered as is, with no warranty) and is designed to enable the exchange of encrypted data between z/OS systems that have the Encryption Facility installed and systems running on z/OS and other platforms that need the supported functions. The Encryption Facility for z/OS Client consists of the following:

- The Java-based Client, which can be used on z/OS and any platform that supports Java. The Java-based client supports both the decryption of data that was created on a z/OS system using the Encryption Facility System z format, as well as encryption of data to be sent to a z/OS system where the file will be decrypted using the Encryption Facility System z format. Data that is to be processed using the Java-based client cannot be created using compression.
- Decryption Client for z/OS. The Decryption Client for z/OS is supported on z/OS systems only. The Decryption Client for z/OS supports decryption of data that was created on a z/OS system using the Encryption Facility System z format. Data that is to be processed using the Decryption Client for z/OS can be created using compression. The Decryption Client does not support data encryption for the return trip. This option might have performance benefits and require less media for exchange purposes but does not allow your business partner to return the data to you in an encrypted format.

Figure 7-40 on page 417 shows several use cases of the Encryption Facility for z/OS. It shows how the Encryption Facility for z/OS Client can run on distributed platforms and also how an OpenPGP-compliant system (distributed platform or not) can encrypt or decrypt data to or from the Encryption Facility for z/OS.

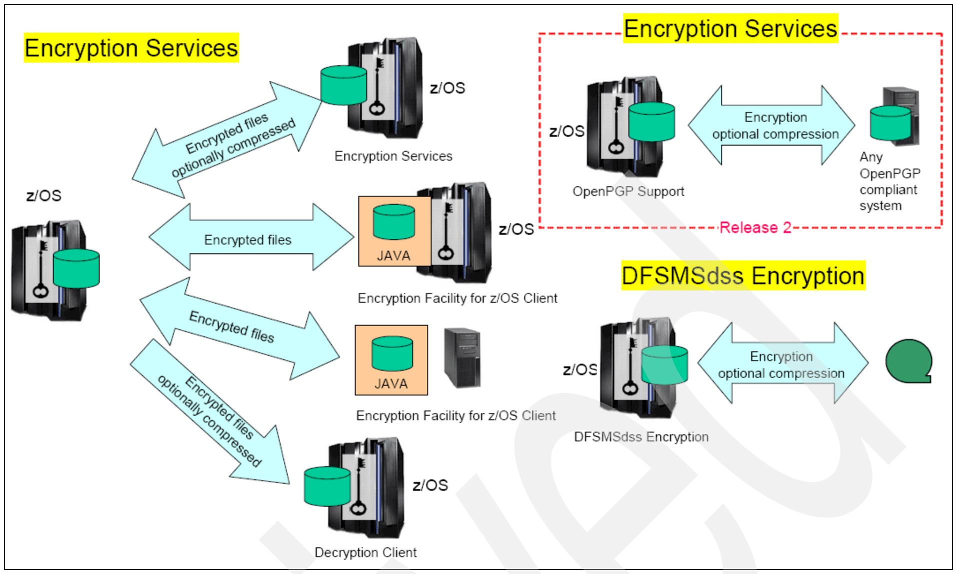

Encryption Services
z/OS
Encrypted files optionally compressed
z/OS
Encryption Services
Encrypted files
JAVA
z/OS
Encryption Facility for z/OS Client
Encrypted files
JAVA
Encryption Facility for z/OS Client
Encrypted files optionally compressed
z/OS
Decryption Client
Encryption Services
z/OS
Encryption optional compression
OpenPGP Support
Any OpenPGP compliant system
Release 2
DFSMSdss Encryption
z/OS
Encryption optional compression
DFSMSdss Encryption

Figure 7-40 Central encryption services with the Encryption Facility for z/OS

With Encryption Facility for z/OS V1.2 the Encryption Services feature has been enhanced to support the OpenPGP standard, RFC 2440. OpenPGP is a standard protocol for ensuring the integrity of data that can be exchanged between trusted partners. It defines the following requirements and suggested practices for data integrity:

- Digital signatures for partner authentication to help ensure that a transferred message has been sent by the party claiming to have sent the message (non-repudiation).
- Data encryption using a randomly generated symmetric session key. The randomly generated session key is encrypted with public key or passphrase-based encryption and prefixed to the encrypted data.
- OpenPGP certificates for the exchange of key information that can provide the data integrity service.

Central database encryption services

The software for this solution runs on z/OS. This solution provides central database encryption service for both IMS and DB2 for z/OS.

IBM Data Encryption for IMS and DB2 databases is a data encryption tool for both IMS and DB2 for z/OS databases. It runs as an exit. The exit code invokes the System z cryptographic hardware to encrypt data for storage and decrypt data for application use. Thus the data is not stored on disk un-encrypted; it is encrypted on the fly before being written to disk.

It is a single tool for both IMS and DB2 databases. The granularity of this encryption is at the IMS segment level or the DB2 table level. It is possible to have different encryption exits for different segments or tables. For example, in IMS a financial application segment could use one exit and a personnel segment another. Log records and image copies of your data are also encrypted.

This tool requires no changes to the applications. It provides an ISPF front end to tailor and build encryption exits. It is a high performance and low overhead solution that uses existing hardware capabilities (the System z Cryptographic Coprocessor feature). Data Encryption for IMS and DB2 databases is implemented by using the IMS Segment Edit/Compression exit and the DB2 EDITPROC exit. The IMS exits and the DB2 Secure Key exit calls Integrated Cryptographic Service Facility (ICSF) routines to do the actual encryption.

Figure 7-41 shows the components and processing flow for Secure Key data encryption in the DB2 environment.

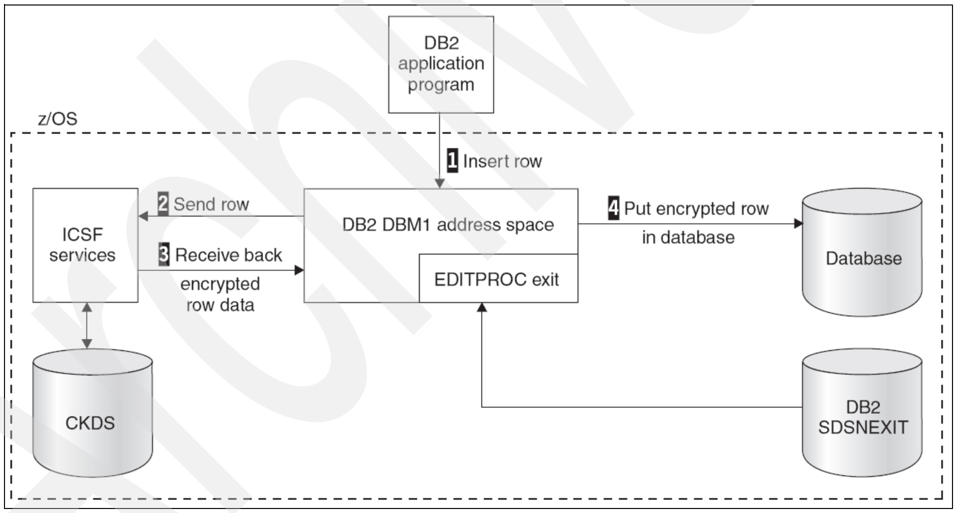


Figure 7-41 IBM Data Encryption for DB2 encryption

For more information, refer to Web site:

`http://www-01.ibm.com/software/data/db2imstools/db2tools/ibmencrypt.html`

Central encryption to tape services with TS1120 and EKM

The software for this solution runs on z/OS, Linux on System z, and on distributed platforms with different capabilities. This solution provides central encryption to tape services for most z/OS and non-z/OS needs.

If tape data on cartridges leaves data centers, the data is no longer protected through RACF or similar access protection mechanisms. Tape Encryption can help fulfill safety regulations. Many governmental agencies are requiring disclosure of security breaches. Industry organizations are also increasing their scrutiny of security procedures.

Important and sensitive data can be protected in various ways. Data can be encrypted by means of special software programs, hardware adapters, facilities, or outside of the device itself, where the data is stored. Encrypting data with software programs takes away processor power and encrypting data with hardware needs additional investment in hardware for the computers.

Tape encryption provides high performance data encryption inside the tape drive itself. Encryption is performed at the tape drive hardware at a speed of up to 100 MBps (un-compressed) and supports encryption of large amounts of tape data for backup and archive purposes. The IBM Tape Encryption solution utilizing IBM System Storage™ TS1120 Tape Drive offers a solution for tape data encryption by offloading encryption tasks from the servers, leveraging existing tape infrastructure incorporated in standard IBM Tape Libraries, and eliminating the need for unique appliance hardware.

You can greatly simplify your Tape Encryption management because the solution provides application-transparent implementations in system-managed environments and library-managed environments.

A Java application named *Encryption Key Manager (EKM)* is necessary to provide keys to the tape drives. It is a flexible solution for tape data key management. It generates and serves data keys to TS1120 tape drives. It stores key encrypting keys' certificates in platform-specific key stores. It runs on heterogeneous platforms such as z/OS, AIX, i5/OS, Linux, Linux on System z, HP, Sun™, and Windows. It supports system-managed and library-managed encryption. The EKM can run on the same or different server than the tape application.

One of the major architecture decisions for a Tape Encryption solution is to define on which platform to run the EKM and which keystore to use for the keys that encrypt the data key on the tapes. On distributed platforms, JCEKS file-based keystore and PKCS#11 hardware keystores are available. On z/OS JCEKS file-based is available; RACF or ICSF are available if you need a higher level of security. Because the data on the tape is very sensitive, having a high security keystore for keys is a natural choice and RACF or ICSF keystores are common solutions.

An EKM on z/OS can be used by both z/OS and distributed environments for central encryption to tape services with very high security keystores.

Figure 7-42 shows a central encryption to tape services solution with TS1120 tape drives and an EKM on z/OS using RACF or ICSF for storing keys in high security keystores. This EKM on z/OS can be used by both the System z and the distributed platforms so it is really a cross-platform central encryption to tape solution. System z platforms use System-Managed Encryption (SME); distributed platforms use Library-Managed Encryption (LME).

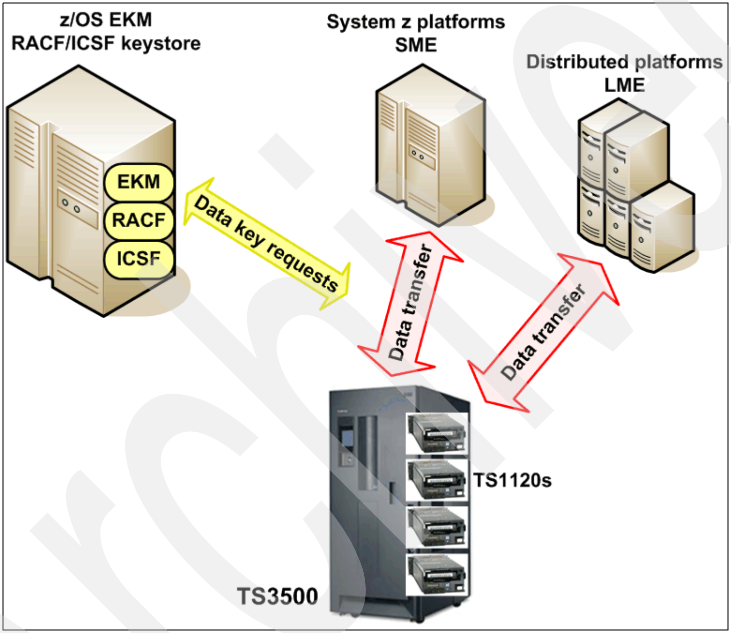


Figure 7-42 Central encryption to tape services with EKM z/OS

7.3.9 Security auditing and compliance solutions

Auditing has taken on increasing importance as compliance to rules has become a key objective of the IT organization. We discuss the following solutions with regard to auditing and compliance:

- Central auditing services with SAF-RACF and SMF
- Central compliance services with Tivoli zSecure
- Central compliance services with TCIM

Central auditing services with SAF-RACF and SMF

The software for this solution runs on z/OS and provides central Auditing services for z/OS security events.

z/OS RACF enables consistent auditing, which is critical for compliance needs. RACF records system events, enabling monitoring of users and their activities. It reports on attempts to perform unauthorized actions. RACF creates SMF records for post processing and provides a Report Writer, XML interface for reporting. The report describes attempts to access RACF-protected resources by user ID, of successful access, or security violations. A common approach avoids auditing integration and compliance challenges posed by inconsistent distributed systems logging. IBM has built auditing capabilities into all of its subsystems to create records which can be used for audit purposes.

RACF maintains the date, time, and number of times users enter the system and the number of times a resource was accessed by any one user. RACF writes security logs when it detects unauthorized attempts to enter the system, and to access RACF-protected resources or commands.

The RACF component gets its input from SMF type 80 records. Type 80 records are produced during RACF processing. RACF writes a type 80 record when one of these events occurs:

- Unauthorized attempt to enter the system.
- Authorized access or unauthorized attempt to access RACF-protected resources.
- Authorized or unauthorized attempt to modify profiles on a RACF database.
- Successful or unsuccessful partner LU verification.

RACF writes one record for each event. The contents of records can be listed to detect potential threats. This is very helpful in auditing and in tracing activity by a user across workloads. The SMF data can be unloaded and the RACF report writer can be used. SMF data can be viewed directly, used as input for user-written programs, or uploaded to database managers such as DB2.

The *RACF report writer* lists the contents of SMF in an easy to read format. This helps with compliance with auditing requirements. For example, with SOX, you probably want to check access and segregation of duties. Other subsystems also write RACF records so that there is a cohesive view of auditing.

The *RACF Data Security Monitor (DSMON)* enables you to verify the basic system integrity and data security controls. RACF auditors can use the DSMON reports to evaluate the level of security at the installation and to compare the actual level of security at an installation with the planned level of security. The Data Security Monitor (ICHDSM00, or usually called DSMON) is a batch program that allows authorized users to obtain a set of reports that provide information about the current status of the installation's data security environment. These reports help to check the initial steps you took to establish system security and make additional security checks periodically. DSMON can be used for a checkup of the system and the security configuration on a regular basis.

The *RACF SMF Data Unload Utility (IRRADU00)* enables installations to create a sequential file from the security-relevant audit data. The sequential file can be used in several ways: viewed directly, used as input for installation-written programs, manipulated with sort/merge utilities, output to an XML-formatted file for viewing on a Web browser, or uploaded to a database manager (for example, DB2) to process complex inquiries and create installation-tailored reports.

Tivoli Decision Support for z/OS along with the RACF Database Unload Utility (IRRDBU00) can collect, organize in DB2 database, and convert raw standard systems security data into business-relevant information that can help improve operational planning, cost management, responsiveness, and decision-making processes in an organization. Tivoli Decision Support for z/OS generates customized reports for communicating and exchanging valuable security information between different departments in an enterprise. Tivoli Decision Support for z/OS has a modern Web browser interface for report generation and publishing. *Tivoli Enterprise Portal (TEP)* along with the *Business Intelligence Reporting Tool (BIRT)* provides all the standard reporting Tivoli Decision Support for z/OS has provided for years in a modern format. Advanced customization capabilities allow for standard and specialized reports to be created on line.

Central compliance services with Tivoli zSecure

The software for this solution runs on z/OS and provides central Auditing and Compliance services for z/OS security events.

The IBM Tivoli zSecure suite is a mainframe security administration, compliance, and audit family of solutions. zSecure suite components allow for administration and policy enforcement of RACF security and monitoring, auditing, and alerting of RACF, ACF2, and Top Secret-based mainframes.

One component of the Tivoli zSecure suite is *Tivoli zSecure Audit*. It is a compliance and audit solution that enables you to automatically analyze and report on security events and detect security exposures. It provides the monitoring and compliance reporting necessary to handle the many regulations that exist in a modern business climate. It provides the capability to report all transactions, detect exposures, and execute status audit. It creates a transparency of the definitions that control the system security. After auditing and analyzing the z/OS operating system, Tivoli zSecure Audit prioritizes and highlights security concerns. Problems are ranked by audit priority, with a number indicating the relative impact of a problem.

Tivoli zSecure Audit can analyze SMF data from the live SMF data sets or from extracted SMF data on tape or disk. The SMF analysis component supports more than 40 standard z/OS SMF record types and includes specific auditing functions for RACF, ACF2, TSS, DB2, and UNIX. Tivoli zSecure Audit includes a powerful system integrity analysis feature. Reports identify exposures and potential threats based on intelligent analysis built into the system.

Tivoli zSecure Audit can automatically send security information from the mainframe into *Tivoli Compliance Insight Manager (TCIM)* for easy company-wide inclusion of the mainframe in audit and compliance reports.

Another component of the Tivoli zSecure suite is *Tivoli zSecure Command Verifier*. This is a policy enforcement solution that enforces compliance to company and regulatory policies by preventing erroneous commands. It helps reduce the risk of security breaches and failed audits caused by internal errors and non-compliant commands. It supports policy definitions to provide mandatory and default values for which RACF does not provide appropriate defaults. It automatically verifies command keywords against specified policies as soon as a RACF command is issued from TSO, ISPF, batch jobs, or an operator console.

Tivoli zSecure Command Verifier can easily discover when a change to a profile was made and which administrator issued a particular command (audit trail of recent RACF commands within the profile). Also it can grant users granular access to specific commands they would normally be unable to access.

Also in the Tivoli zSecure suite, *Tivoli zSecure Alert* allows detection and reporting of security events and exposures on z/OS, DB2, UNIX System Services, RACF, ACF2, and TopSecret. It possess a threat knowledge base with parameters from your active configuration and a broad range of monitoring capabilities. This helps isolate relevant attack patterns, and detect multiple types of attacks and configuration threats, including those external to the event log (SMF record).

Tivoli zSecure Alert raises awareness of violations and intrusions, while Tivoli zSecure Command Verifier monitors all commands being submitted, preventing inappropriate operations.

Tivoli zSecure Alert can automatically send security information to Tivoli Compliance Insight Manager (TCIM), Tivoli Security Operations Manager, and network and enterprise consoles using SNMP.

Central compliance services with TCIM

The software for this solution runs on distributed platforms and provides central Compliance services for z/OS, Linux on System z, and distributed security events.

Tivoli Compliance Insight Manager (TCIM) connects the mainframe to an enterprise compliance dashboard for reporting across applications, databases, and operating systems. It provides visibility into the organization's security compliance posture through automated, enterprise-wide user activity monitoring. It includes dashboard views and reporting to help measure security posture and respond to auditors' requests.

It allows *Privileged User Monitoring and Audit (PUMA)* on databases, applications, and servers, all powered by a comprehensive log and audit trail collection capability. This unobtrusively monitors and reports on privileged user activities, allowing the administrators to perform their jobs and supporting strong controls over user access. Its scalable log collector helps ensure the reliable and verifiable collection of native logs from virtually any platform, including syslog and Simple Network Management Protocol (SNMP) logs, and almost any security log type, including operating systems, databases, and security devices.

Through *Tivoli Compliance Insight Manager Enabler for z/OS*, z/OS events and alerts can be managed centrally. Tivoli Compliance Insight Manager can use the event data that is created through normal z/OS SMF processing. It copies this data to a file that is stored in UNIX System Services and then passes the data to the server. The originating SMF data is not deleted or moved, the data is only copied. This maintains other processes that use this data (such as zSecure solutions) to report on specific data and events outside of what TCIM accomplishes in the mainframe environment.

Tivoli Compliance Insight Manager integrates with Tivoli Identity Manager, Tivoli Access Manager, and Tivoli Security Operations Manager.

7.4 Service Assurance management

As enterprises increasingly seek to measure their performance from the perspective of their customers, they run into a number of challenges in the way they manage SLAs. There is a temptation to over-provision resources to meet SLAs, or to implement a time-consuming, labor-intensive and inaccurate method of service level management to measure availability and response time for each resource. To avoid these expensive pitfalls, the organization needs to manage performance and availability of the IT infrastructure effectively. Therefore, we need a solution that lets us define SLAs and generate SLA reports easily, and that can help us to generate early warnings and proactively resolve potential problems before service levels are affected.

7.4.1 Key architectural decisions with respect to service assurance management

Consider the following key issues when planning a service assurance management solution:

- What are the requirements for the management of the ongoing health of services, processes, transactions, and business activities?
- Are the current service assurance model and strategies effective? This model could be implemented in several phases such as planning, implementing, and administrating.
- What service assurance management techniques and measurements will be most useful in your specific environment? Should you implement service intelligence via real-time service scorecards, Key Performance Indicators (KPIs) and SLA tracking, and so forth?

7.4.2 Service level management solution

Service level management enhances the availability and performance management that we discussed earlier in this chapter. It helps the IT organization map its activities to the real needs of the business. One of the critical planning areas is setting appropriate objectives and targets. If the business accepts a Service Level Objective (SLO) without knowing what it will take to reach that objective, it will very likely fail. The next step is implementation, which shifts from a technology focus to a business service focus. That is, we need to define and distinguish between services based on their relative importance to the business and send appropriate information when the service level is not met. The last step is to map the relationship between the IT services and the IT infrastructure as well as the administration once the solution is implemented. See Figure 7-43 on page 426.

In this section, we present a number of Tivoli products that are available for our service level management solution:

- IBM Tivoli Change and Configuration Management Database (CCMDB)
- IBM Tivoli Application Dependency Discovery Manager (TADDM)
- IBM Tivoli Business Service Manager (TBSM) for z/OS (or Linux on System z)
- IBM Tivoli Service Level Advisor (TSLA)

- IBM Tivoli Data Warehouse
- IBM Tivoli Composite Application Manager for Response Time Tracking/SOA/WebSphere Application Server (ITCAM for RTT/SOA/WebSphere Application Server)
- IBM Tivoli Monitoring for distributed systems (ITM)
- IBM Tivoli OMEGAMON XE for MVS, CICS, IMS, DB2, Messaging, WebSphere Application Server, and so forth
- IBM Tivoli NetView for z/OS Discovery Library Adapter (DLA) for CICS, IMS, DB2, MQ and WebSphere Application Server

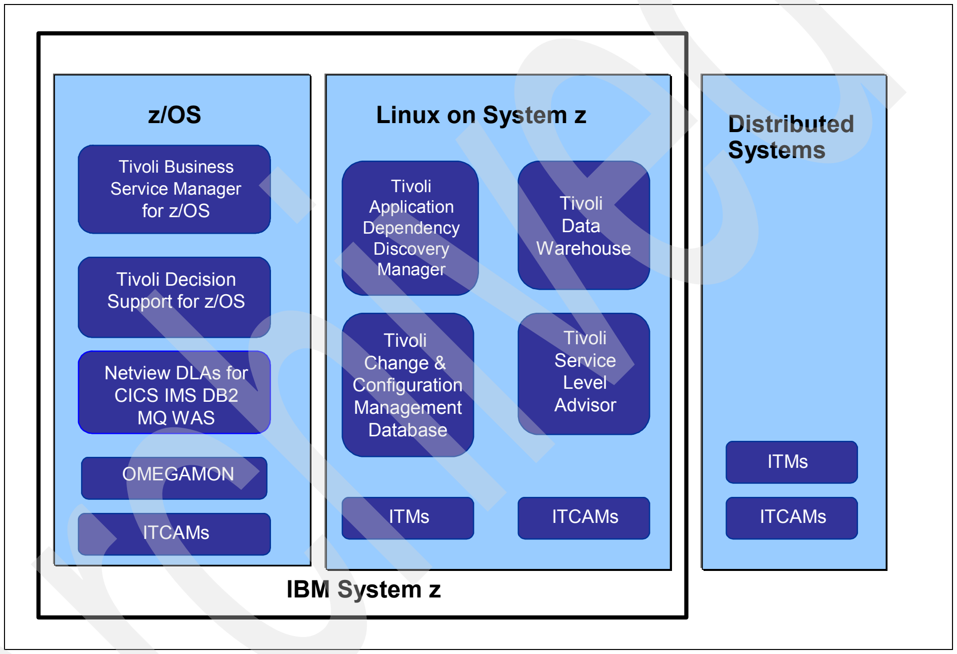

z/OS
Tivoli Business Service Manager for z/OS
Tivoli Decision Support for z/OS
Netview DLAs for CICS IMS DB2 MQ WAS
OMEGAMON
ITCAMs
Linux on System z
Tivoli Application Dependency Discovery Manager
Tivoli Data Warehouse
Tivoli Change & Configuration Management Database
Tivoli Service Level Advisor
ITMs
ITCAMs
Distributed Systems
ITMs
ITCAMs
IBM System z

Figure 7-43 Service level management solution

Most of the products identified here might be installed and running already for other IT Service Management purposes; if so, they also have been discussed previously. For instance, all the monitoring components were covered in the Availability and Performance management section. Therefore, only the core components of our service level management solution will be described in this section.

IBM Tivoli Change and Configuration Management Database (CCMDB)

IBM Tivoli Change and Configuration Management Database (CCMDB) provides an enterprise-ready configuration management database and platform so that we can standardize and share information that helps integrate people, processes, information, and technology. We use CCMDB to identify and manage resources in the context of the services. We can manage the desired states of configuration items, applications, and service configurations. We can create, assign, monitor, notify, act upon, and report on change requests and configuration items using the provided Portal interface. CCMDB leverages the advanced discovery capabilities of *IBM Tivoli Application Dependency Discovery Manager (TADDM)* to help automatically create and maintain application infrastructure maps that include complete run-time dependencies, deep configuration values, and accurate change histories. CCMDB discovers information and the relationships between the IT components. Through the use of discovery library adapters, for example, NetView DLAs, data can also be consolidated from any data source. Once the data is consolidated, information can be visualized to assist in change impact assessment or incident analysis. We can drill down to see detailed configuration information, assess key details about Configuration Items (CIs) and their relationships, including information from other management tools, assess and anticipate the business impact of changes and problems, and so forth. CCMDB comes with an out-of-the-box change process to help reduce scheduling conflicts and problems caused by changes. We can use change templates to drive different change flows depending on requirements. As a result, we can ensure the configuration data is current through change and configuration process management with CCMDB.

IBM Tivoli Application Dependency Discovery Manager (TADDM)

IBM Tivoli Application Dependency Discovery Manager (TADDM) is a major component of the IBM Tivoli Change and Configuration Management Database (CCMDB), providing the core services for discovery, configuration auditing, reconciliation, and federation. We need to know what Configuration Items (CIs) we have, how the CIs are configured and changing over time, and determine if the CIs are compliant. TADDM is a robust application mapping and discovery tool that gathers an inventory of applications, configurations, and dependencies and generates detailed reports, for example, auditing reports. Using the supplied discovery scanning profiles we can capture the infrastructure configurations and application dependencies. This information can be organized into application maps and views, for example, an enhanced depth of discovery for IBM z/OS configurations, along with additional topology views and new Sysplex graph. TADDM also provides a query and reporting facility that helps address internal configuration auditing requirements. It can be integrated with other IBM Tivoli products such as Tivoli Business Service Manager (TBSM), IBM Tivoli Monitoring (ITM), ITCAM, and CCMDB, providing the core services for discovery and configuration auditing.

Figure 7-44 illustrates one possible configuration.

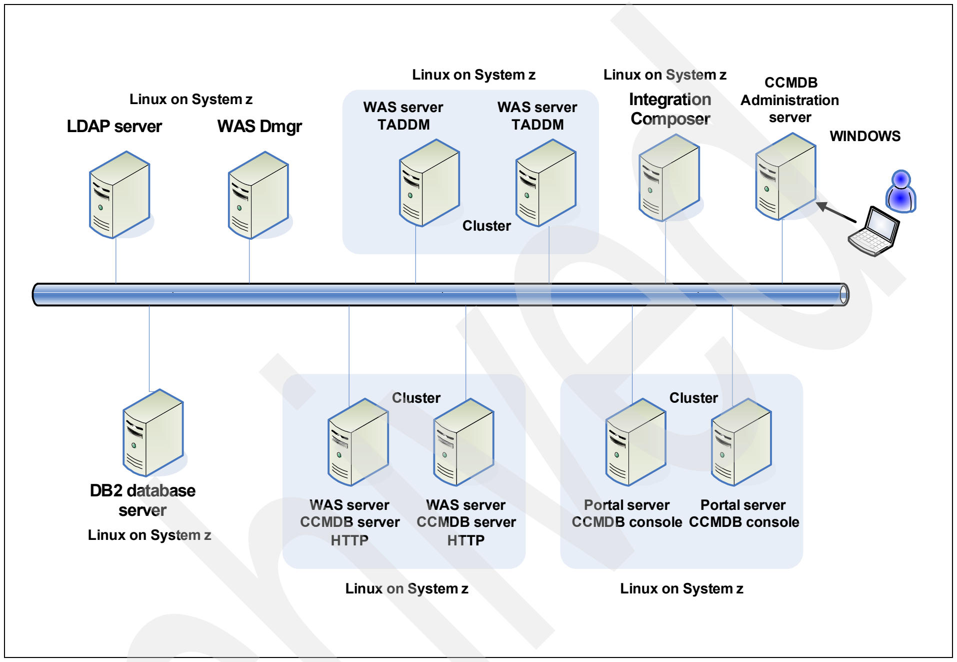


Figure 7-44 CCMDB and TADDM architecture

IBM Tivoli Business Service Manager (TBSM)

IBM Tivoli Business Service Manager (TBSM) for z/OS (or Linux on System z) provides end-to-end availability management across z/OS and distributed systems. TBSM can manage both mainframe and distributed systems and is able to view IT resources in the context of critical business services. Its flexible Web GUI allows information to be accessed from anywhere at any time. Integrated with the Tivoli Service Level Advisor, TBSM can dynamically track operational, business, and customer SLAs and KPIs. It also can link to CCMDB, TADDM, OMEGAMON XE, NetView DLAs, and other Tivoli monitoring components to allow service model changes in the dynamic IT environment. It auto-populates service dependencies and tracks changes in the IT infrastructure. As a result, it discovers and builds graphical views of applications automatically and keeps the business system view up-to-date by avoiding the problem of manual entry leading to obsolete information displays.

The out-of-the-box service level tracker defines and tracks compliance with service level data in real time, such as:

- SLA state
- Percentage of time a service has been up
- Total downtime for an SLA period
- Time left until an SLA is breached
- Total cost of downtime for a given SLA period

In addition, TBSM tightly integrates with *IBM Tivoli Service Level Advisor* to augment its SLA abilities, achieving a full-featured SLA management solution. Because Tivoli Service Level Advisor integrates not only with TBSM but also with performance and availability measurement functions from other ITM and ITCAM software, or any other source that feeds IBM Tivoli Data Warehouse, more SLA information can be provided, such as for:

- Provisioning SLAs that are meaningful to the business across all applications
- Creating and reviewing historical data for SLAs
- Automating production and timely delivery of SLA reports to executives, operations staff, and customers
- Gaining early warnings about trends that could result in breached SLAs
- Providing the intelligence needed for ongoing SLA planning and correction

IBM Tivoli Service Level Advisor (TSLA)

IBM Tivoli Service Level Advisor (TSLA) provides an easy to use wizard to define and build SLAs. Tivoli Service Level Advisor helps to write SLAs that manage transaction performance and availability. The SLAs can encompass data that pertains to mainframe resources, distributed resources, resource utilization, and service desk information. All of the extensive systems management data that is available within the Tivoli Data Warehouse (including non-Tivoli data) can be utilized in SLA creation. TSLA supports the ITIL® best-practice standards for IT and enables you to implement an ITIL-based service level management process. It proactively predicts when Service Level Agreement (SLA) violations are likely to occur and then takes corrective action to avoid the violation, saving both time and costs. TSLA integrates service-level data with availability data on the Tivoli Business Systems Manager (TBSM) executive dashboard to increase executive knowledge of IT as it relates to business objectives. It utilizes a patent-pending trend analysis algorithm for predictive trend analysis. TSLA feeds data to the Tivoli Data Warehouse to consolidate and report on systems management data. TSLA can automatically generate Web-based, graphical SLA reports to meet the business requirements such as SLA summary status dashboard, SLA listing by customer, SLA business activities summary, and so forth. In addition, reports can be exported to Microsoft Excel® or a PDF format.

Figure 7-45 shows a representative configuration of TBSM and TSLA.

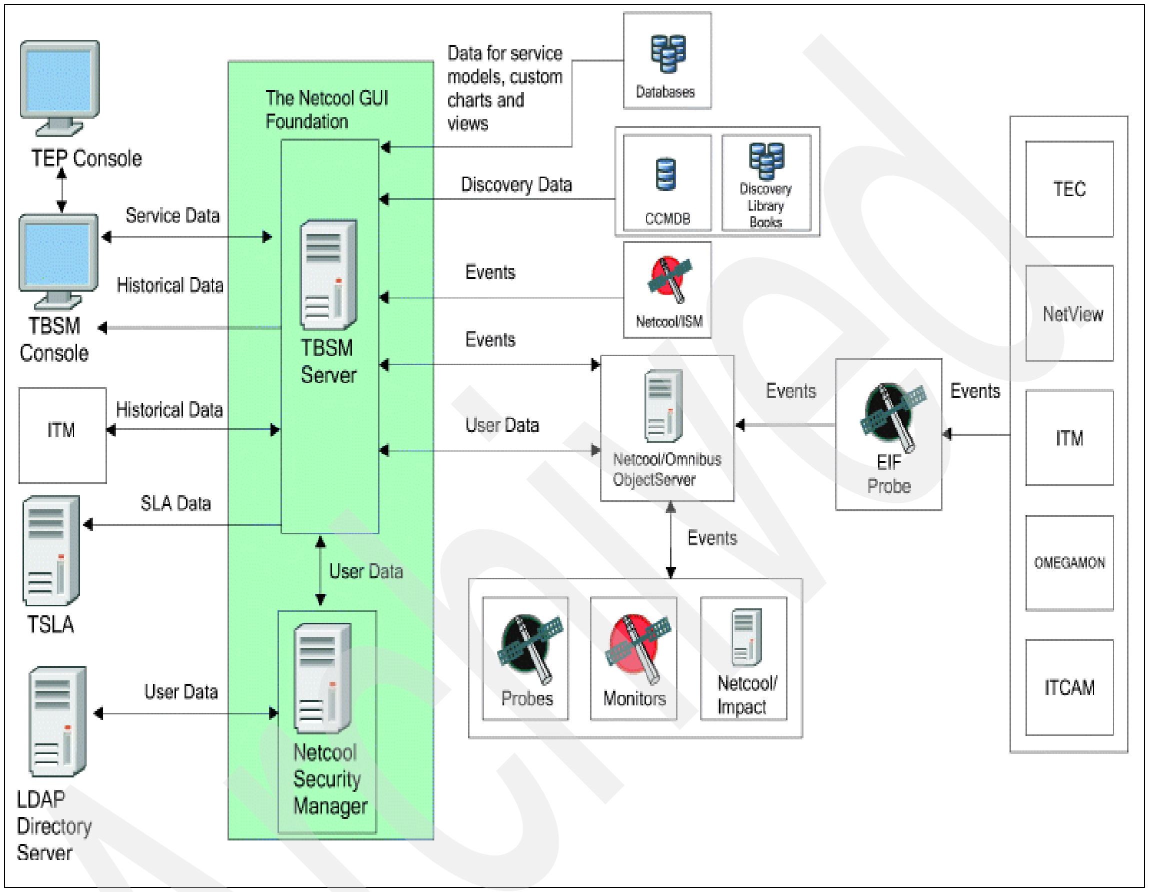


Figure 7-45 TBSM and TSLA architecture

7.5 Capacity management

Even though many large companies have capacity management tools, experts, and organizations in place, capacity-related business service outages still exist. This could be due to difficult or misunderstood processes, or the management of IT capacity could be ineffective. Sometimes rapid changes are made to the infrastructure to solve availability problems or to add new application functionality. These changes tend to be made on a project-by-project basis, without going through a formal, business-driven capacity planning process. Lacking a comprehensive overview of the entire infrastructure, organizations often dedicate too few resources, which causes performance bottlenecks, or too many resources, which creates waste and unnecessary costs. Even when experts are involved, they might be left to guess about what data to collect, how much capacity is really available, and how to effectively balance enterprise-wide resources and workloads.

The goal of capacity management is to ensure that the capacity of IT resources matches the demands of business for IT services in a timely and cost effective way. The capacity management solution will help to establish a disciplined approach, using capacity planning tools regularly and in a repeatable pattern, and provides a formal way to gather and analyze data, model, and approve all capacity management activities. Therefore, it leads to fewer capacity-related business service outages and helps the organization achieve better infrastructure utilization and fewer performance problems.

7.5.1 Key architectural decisions with respect to capacity management

Consider the following key issues when planning a capacity management solution:

- Are there any misunderstood processes and is current IT capacity management effective?
- What processes need to be enhanced to support the capacity management scenarios, including sizing a new application, tuning an existing deployment, and monitoring a deployed infrastructure?
- What gaps exist in the IT infrastructure for capacity management processes and what assessments must be conducted?
- What tooling and planning is required to implement the capacity management solution?
- Are the necessary skills available within the organization or are experts from the outside required?

7.5.2 Capacity management solution

The capacity management solution supports a process model that can be based on ITIL frameworks or the IBM RUP process. It establishes a disciplined approach using capacity planning tools and provides a formal way to gather data, analyze it, model, and approve all capacity management activities. The solution is capable of automating workflows for multiple types of capacity management requests. Figure 7-46 on page 432 shows the capacity management solution using *IBM Tivoli Capacity Process Manager (TCPM).*

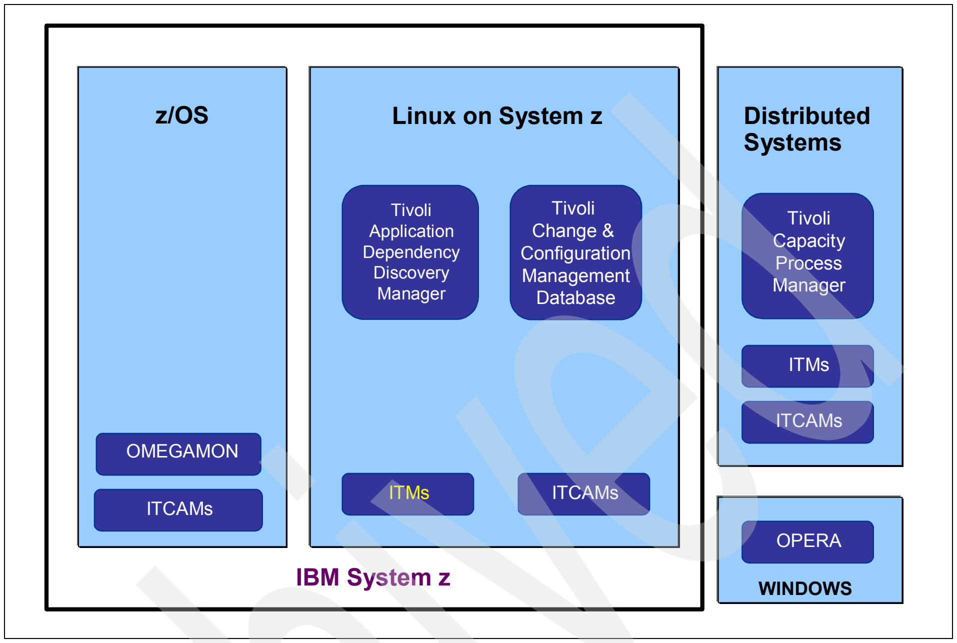

z/OS
OMEGAMON
ITCAMs
Linux on System z
Tivoli Application Dependency Discovery Manager
Tivoli Change & Configuration Management Database
ITMs
ITCAMs
IBM System z
Distributed Systems
Tivoli Capacity Process Manager
ITMs
ITCAMs
OPERA
WINDOWS

Figure 7-46 Capacity management solution

IBM Tivoli Capacity Process Manager (TCPM) is fully integrated with IBM Tivoli Change and Configuration Management Database (CCMDB) and designed to work with the existing operational management products such as IBM Tivoli Monitoring (ITM). See "Service level management solution" on page 425 for a description of CCMDB. TCPM has a set of customizable process templates that support common capacity management scenarios, such as sizing a new application, tuning an existing deployment, and monitoring a deployed infrastructure. It provides a library of customizable, best-practice workload profiles for capturing and modeling common applications, such as online shopping and batch operations. A library of task-specific expert advice guides the nonspecialist staff through the capacity management process. The process manager task flow includes two major customizable templates that cover the most common capacity management processes:

- Sizing new applications
- Monitoring existing applications

The template for sizing a new application helps to size new business applications in a timely fashion so that the right infrastructure capacity is available to meet the business needs once an application goes live. It captures application characteristics, critical performance and capacity metrics, and creates an application profile. The capacity modeling tool provides a capacity estimate of the resources the new application will need to meet service level targets for the projected workload. After the profile is created, TCPM will send a profile to the management personnel for vetting and approval.

TCPM can also monitor existing applications and helps you manage capacity of existing applications and services. You can then determine if a capacity issue is arising. The performance and resource utilization of the target applications are calculated based on the monitoring results. You can speed up this workflow by using IBM Tivoli Enterprise Portal in conjunction with a monitoring tool, such as IBM Tivoli Monitoring (ITM), IBM Tivoli Composite Application Manager (ITCAM) or IBM Tivoli OMEGAMON monitors. You can use capacity and performance-related reports from Tivoli Enterprise Portal software to compare an application and its performance to preset objectives and to look for current or potential bottlenecks. Tivoli Capacity Process Manager (TCPM) integrates fully with Microsoft Excel, and can be used in conjunction with *On Demand Performance Advisor (OPERA)*, a capacity estimator tool available from IBM *Open Process Automation Library (OPAL)* as a no-charge download.

By using Tivoli Capacity Process Manager to implement, enforce, and track all capacity management activities, you can avoid potentially wasteful or insufficient upgrades and changes to the IT infrastructure. Figure 7-47 on page 434 shows a representative infrastructure with TCPM and other Tivoli solutions.

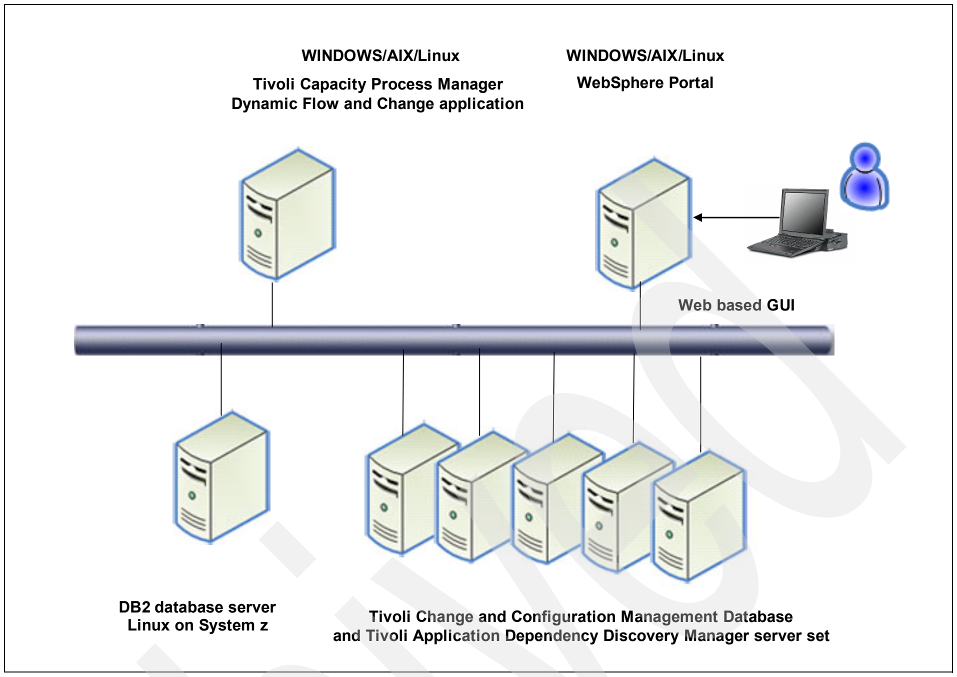

WINDOWS/AIX/Linux
Tivoli Capacity Process Manager
Dynamic Flow and Change application
WINDOWS/AIX/Linux
WebSphere Portal
Web based GUI
DB2 database server
Linux on System z
Tivoli Change and Configuration Management Database
and Tivoli Application Dependency Discovery Manager server set

Figure 7-47 Tivoli Capacity Process Manager architecture

Note: At the time of writing, TPCM is available for Linux but we were unable to confirm if the solution is supported on Linux on System z.

7.6 Service continuity management

Planning to ensure business continuity plays an important role in today's business activities. The supporting IT services must be capable of handling unexpected major service outages so that the business activities are not affected. The Service continuity solution ensure that specific IT services will support business requirements in the event of a disruption to the business, based on a committed recovery schedule. Service continuity is part of the business continuity management process that includes anticipating incidents that might cause discontinuity in critical business functions and planning how to handle such incidents.

7.6.1 Key architectural decisions with respect to service continuity management

Consider the following key questions when planning a service continuity management solution:

- Is the IT services recovery and continuity part of the enterprise business continuity plan?
- Are there services recovery strategies in place? The most commonly used recovery strategies are Do Nothing, Manual Work around, Reciprocal Arrangement, Gradual Recovery, Intermediate Recovery, Fast Recovery, and Immediate Recovery.
- What is the existing recovery plan and what are the options for service outage? Service outage could be a single service or multiple services. The most commonly used options might be to restore the failed services and the underlying IT components on the same IT facility or separate dedicated IT facilities.
- What is the existing recovery plan and what are the options when a major disaster happens? Major disaster can imply the failed services cannot be restored in the same IT facility or it is not possible to restore services within the agreed IT services schedule. The most commonly used options are to make use of dedicated facilities, or third party facilities shared by multiple businesses. What topology best suits the business requirements?
- What are the testing requirements for the service continuity solution?

7.6.2 Service continuity automation solution

The service continuity automation solution should be comprehensive and include more then just the technology layers: it should incorporate the people and process elements as well. The solution should effectively respond to incidents and events that can significantly impact business operations and also be able accommodate ongoing changes in the environment. The solution should also allow for meaningful testing, with minimal disruptions to the production environment, and mitigate the risk of an incomplete or insufficient business continuity plan.

Figure 7-48 on page 436 shows a service continuity automation solution from a management view.

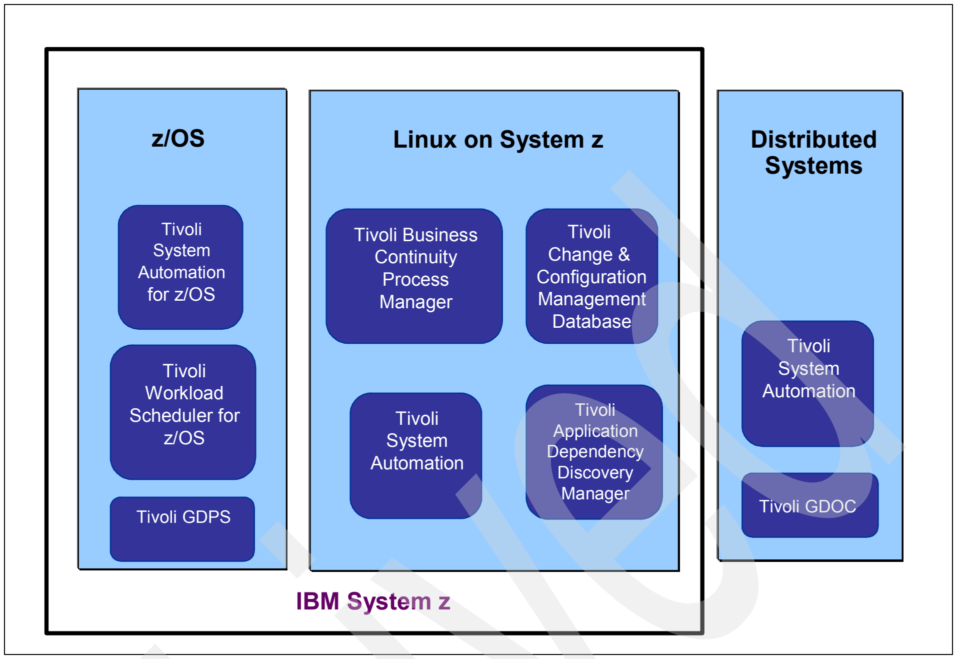

z/OS
Tivoli System Automation for z/OS
Tivoli Workload Scheduler for z/OS
Tivoli GDPS
Linux on System z
Tivoli Business Continuity Process Manager
Tivoli Change & Configuration Management Database
Tivoli System Automation
Tivoli Application Dependency Discovery Manager
Distributed Systems
Tivoli System Automation
Tivoli GDOC
IBM System z

Figure 7-48 Service continuity automation solution

IBM Tivoli Business Continuity Process Manager (TBCPM) is based on the infrastructure provided by IBM Tivoli Change and Configuration Management Database (CCMDB). No additional hardware is required besides what is needed to deploy CCMDB. TBCPM can run on Linux on System z. IBM Tivoli Business Continuity Process Manager is installed using the Process Solution Install (PSI) Client that is provided with CCMDB. You can install Business Continuity Process Manager through the PSI user interface (an installation wizard), or using the PSI command line interface. It works together with Tivoli System Automation for z/OS and Tivoli Workload Scheduler for z/OS to provide an automatic service continuity solution.

Figure 7-49 on page 437 shows a representative infrastructure with TBCPM and other Tivoli products.

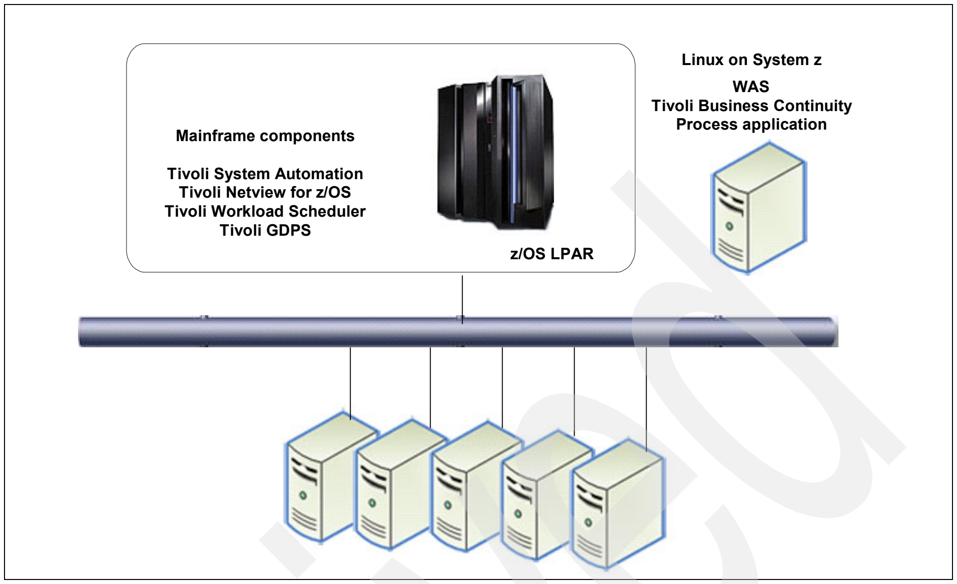


Figure 7-49   Tivoli Business Continuity Process Manager architecture

The IBM Tivoli System Automation family provides a single point of control and management for critical applications and IT services across distributed systems and z/OS. IBM Tivoli System Automation for z/OS was covered in “System automation solution” on page 359 and therefore we do not further discuss it in this section.

Tivoli Workload Scheduler (TWS) for z/OS enables automation, planning, and controlling of processing of z/OS workloads, such as backup and restore jobs, and invoking of system commands through Tivoli System Automation for z/OS. TWS was covered in “A workload management solution” on page 360 and therefore we do not further discuss it in this section.

IBM Tivoli Change and Configuration Management Database (CCMDB) is a key element of IT Service Management. It provides an enterprise-ready configuration management database and platform upon which you can standardize and share information that helps integrate people, processes, information, and technology. We use CCMDB to automatically discover information about the IT infrastructure and the relationships between the components within it. Through the use of discovery library adapters, data can also be consolidated from others servers. CCMDB runs on distributed systems including Linux on System z. Data is easily accessed via an API and a GUI. An open standards-based Business Intelligence Reporting Tool (BIRT) based on Eclipse technology can be used to report on any attribute or data item contained within the database. CCMDB leverages the discovery capabilities of IBM Tivoli Application Dependency Discovery Manager

(TADDM) to help automatically create and maintain application infrastructure maps that include complete run-time dependencies, deep configuration values and accurate change histories. See Figure 7-44 on page 428 for CCMDB and TADDM architecture.

7.6.3 Geographically Dispersed Parallel Sysplex (GDPS) solution

When planning for business resiliency, it is necessary for the business and its data to be able to survive a site failure. *Geographically Dispersed Parallel Sysplex (GDPS)* is a multi-site or single-site end-to-end application availability solution that provides the capability to manage remote copy configuration and storage subsystems (including IBM TotalStorage® Enterprise Storage Server®), to automate Parallel Sysplex operation tasks and perform failure recovery from a single point of control. GDPS consists of a set of disaster recovery and continuous availability solutions. It includes *Peer-to-Peer Remote Copy (PPRC)*, also referred to as *Metro Mirror*, *eXtended Remote Copying (XRC),* also referred to as *z/OS Global Mirror*, or Global Mirror disk replication architectures. GDPS supports z/OS and Open System data, and it is application independent.

GDPS works with Tivoli Business Continuity Process Manager and provides integration of GDPS configuration data in a change management database as well as integration of GDPS as operational manager in a workflow support for Business Continuity.

Remote copy technologies

FlashCopy® copies full volumes of data within the storage subsystem. It takes only a few seconds to establish the FlashCopy relationships for tens to hundreds or more of volume pairs. The copy is then immediately available for both read and write access. The quickness of the FlashCopy operation means it can take multiple FlashCopies of the same volume for use with different applications, for example, data mining operations. Production data can be copied for long-running data mining activities without affecting the production environment.

Global Copy copies data asynchronously over virtually unlimited distances. During the copy, the source volume sends a periodic, incremental copy of updated tracks to the target volume, instead of sending a constant stream of updates. This causes less impact to application writes for source volumes and less demand for bandwidth resources, while allowing a more flexible use of the available bandwidth. Global Copy is a recommended solution for remote data copy, data migration, off-site backup, and transmission of inactive database logs, without affecting application performance, which is particularly relevant when implemented over continental distances.

Metro Mirror (PPRC) mirrors disk volumes between two locations up to 300 km apart. It is supported by the IBM Enterprise Storage Subsystem DS8000® family. It is a synchronous copy solution where write operations are completed on both the local and remote site. Metro Mirror (PPRC) provides *zero data loss*, and a Recovery Point Object of zero (RPO=0). The GDPS *HyperSwap™* function, which requires Metro Mirror (PPRC), allows near-continuous availability for disk failure events. However, the distance between the primary and secondary sites will determine the affect on application response time due to its synchronous mirroring characteristics. Detailed analysis and benchmarks might be possible to determine the exact delay.

z/OS Global Mirror (XRC) is a disk copy function only for z/OS and Linux on System z that is based on the System Data Mover (SDM) feature. z/OS Global Mirror copies the data asynchronously from the local site to the remote location without the 300 km limitation. In fact, the distance is virtually unlimited. In addition, the distance between primary and secondary site will not affect the application response time due to its asynchronous mirroring characteristics.

Note: One of the recent enhancements is the capability to run the System Data Mover (SDM) on one of the specialty engines, the zIIP.

Global Mirror (asynchronous PPRC) provides an asynchronous remote disk-to-disk copy function across two sites over a long distance. Global Mirror is based on the two other disk copy features, Global Copy and FlashCopy. Global Mirror writes the host data to the disk subsystem first at the local site, and then data is asynchronously shadowed to the disk subsystem at the remote site. A consistent copy of the data is automatically maintained at the remote site.

Global Mirror allows the replication of data over long distances asynchronously without impacting the application I/O response time. It is transparent and autonomic to the application and it provides a consistent copy of the data at the remote site at all times. Figure 7-50 on page 440 shows the Global Mirroring.

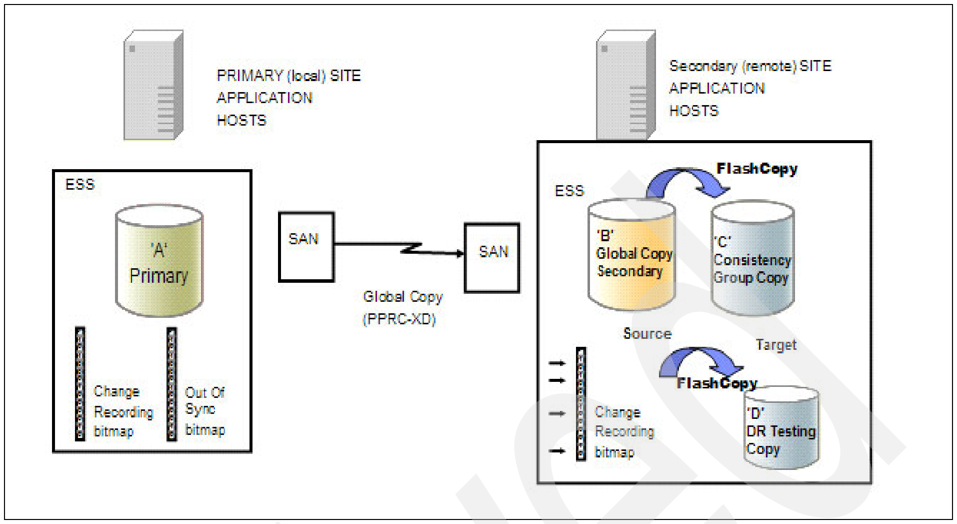


Figure 7-50 Global Mirror

Geographically Dispersed Parallel Sysplex (GDPS) options

Geographically Dispersed Parallel Sysplex (GDPS) solutions are based on the copy technologies described in the previous paragraphs. IBM GDPS is designed to work with Metro Mirror (PPRC), z/OS Global Mirror (XRC), or Global Mirror disk replication protocols. These protocols support the IBM TotalStorage DS6000™ and DS8000 families of disks. Following are several GDPS options (between two locations) offered by IBM:

- GDPS/PPRC, a near Continuous Availability or Disaster Recovery solution across two sites separated by distances up to 100-200 km. The solution is based on Metro Mirror (PPRC), which is a synchronous remote copy.
- GDPS/PPRC HyperSwap Manager, similar to GDPS/PPRC. It provides the ability to transparently switch primary PPRC disk subsystems with the secondary PPRC disk subsystems.
- GDPS/XRC, a Disaster Recovery solution across two sites without the distance limitation between sites. The solution is based on z/OS Global Mirror (XRC), which is an asynchronous remote copy.
- GDPS/Global Mirror, a Disaster Recovery solution across two sites separated by a virtually unlimited distance. The solution is based on the Global Mirror asynchronous PPRC technology which is an asynchronous form of remote copy.
- RCMF/PPRC. a subset of the functions provided by GDPS/PPRC and GDPS/PPRC HyperSwap Manager. Remote Copy Management Facility

(RCMF) for PPRC provides a central point of control to provide a global PPRC configuration awareness.

- RCMF/XRC, a subset of the functions provided by GDPS/XRC and similar to RCMF/PPRC, Remote Copy Management Facility (RCMF) for XRC provides a central point of control to provide a global XRC configuration awareness.

Two mixed GDPS options that could be suitable for a three locations scenario are:

- GDPS Cascaded, a three site solution that provides Continuous Availability across two sites within a metropolitan distance and Disaster Recovery to a third site at a virtually unlimited distance. It combines XRC and Global Mirror. The site with synchronously shadowed disks is used for continuous availability and the asynchronously shadowed disks are used for Disaster Recovery.
- GDPS Multi-target is similar to GDPS Cascaded, but disk copies are multi-targeted to two different sites. The advantage of this option is the asynchronous disk copy can continue to the third site when the second has taken over from the first site if the first site is shut down due to major incident and unable to recover within a short time.

Figure 7-51 on page 442 shows the two mixed GDPS options. GDPS not only provides all the continuous availability benefits of Parallel Sysplex, but also significantly enhances the capability of an enterprise to recover from disasters and other failures, and manages planned exception conditions.

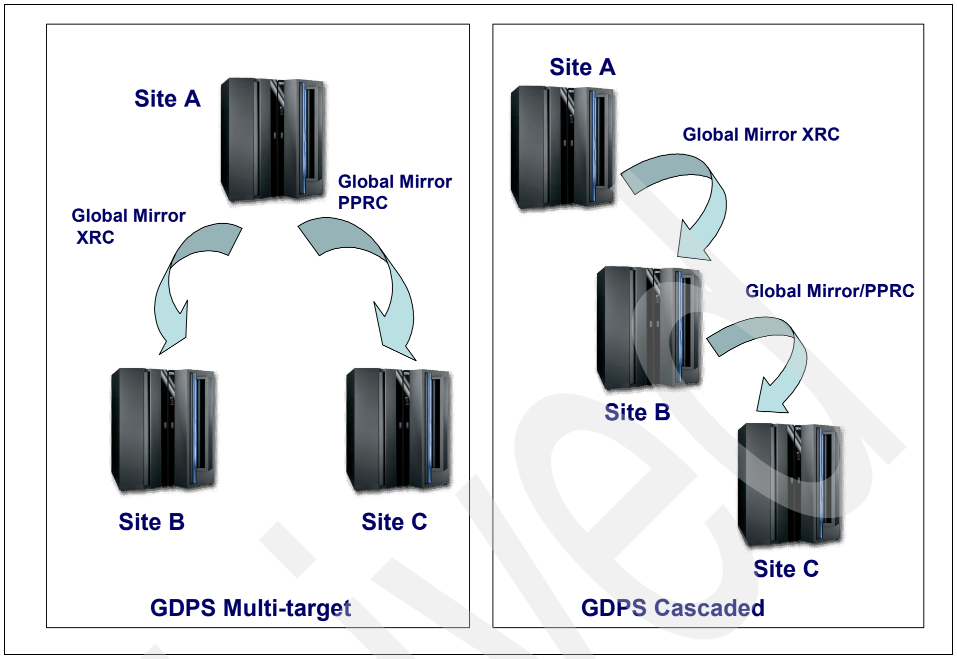


Figure 7-51 GDPS Multi-target and GDPS Cascaded

7.6.4 Globally Dispersed Open Clusters (GDOC) solution

Geographically Dispersed Open Clusters (GDOC) is designed to protect the availability of critical applications running in mixed environments that include IBM AIX-based servers as well as servers based on HP-UX, Sun Solaris™, Linux or Microsoft Windows operating systems. This multivendor solution, which is based on an open systems cluster architecture, provides disaster recovery for data mirrored across two or more sites. A solution typically starts with a cluster of servers at a single site managed by Symantec software to provide high availability. This solution is extended by adding mirrored copies of the data, along with additional servers, at a second, geographically distant site, synchronizing the data between the two sites using one or more data replication solutions. This solution has no System z involvement, so it is not pertinent and not discussed further in this book.

7.7 Asset and financial management

Because IT is evolving to become a service provision solution based on SOA models, it makes IT financial accountability hard to attain. This is made even more difficult to accomplish when IT resources are shared. Virtualization can add layers of complexity to already complex physical IT resources, so that accurately associating costs to users and matching expenses to business use and objectives is difficult or impossible. IT spending becomes essentially unmeasurable and uncontrollable. Therefore, an effective asset and financial management solution is required to address the provision and performance of the shared IT assets. Such a solution will improve IT asset utilization and efficiency, and at the same time, it will help to reduce the total number of assets deployed and lower the maintenance and the management costs.

7.7.1 Key architectural decisions with respect to asset and financial management

Consider the following questions when making architectural decisions regarding an asset and financial management solution:

- What IT resource accounting and charge back processes are required?
- Is the existing measurement of resource usage and charge back effective? IT cost management includes tracking, allocating, and invoicing based on actual resource usage.
- Is an accurate measurement tool available, with the underlying billing processes and Service Level Agreement (SLA) compliance?
- Is the asset and financial management model based on the best practices?

7.7.2 Accounting management solution

In the SOA world, all services are loosely coupled and we need a way to track, allocate, bill, or charge back on actual resources used. The solution will monitor the use of applications, servers, and other IT resources and their complex interactions across distributed and mainframe platforms. We need to justify IT resources and expenses, assessing the cost of IT services and applications, and measuring asset usage in virtualized environments. Therefore, IT needs a solution and product that it can depend upon to help it manage through these changes and others as time goes on without sacrificing the ability to plan, analyze, report, and meet audit requirements on service delivered and at what cost.

Figure 7-52 shows a high level view of the Accounting management solution.

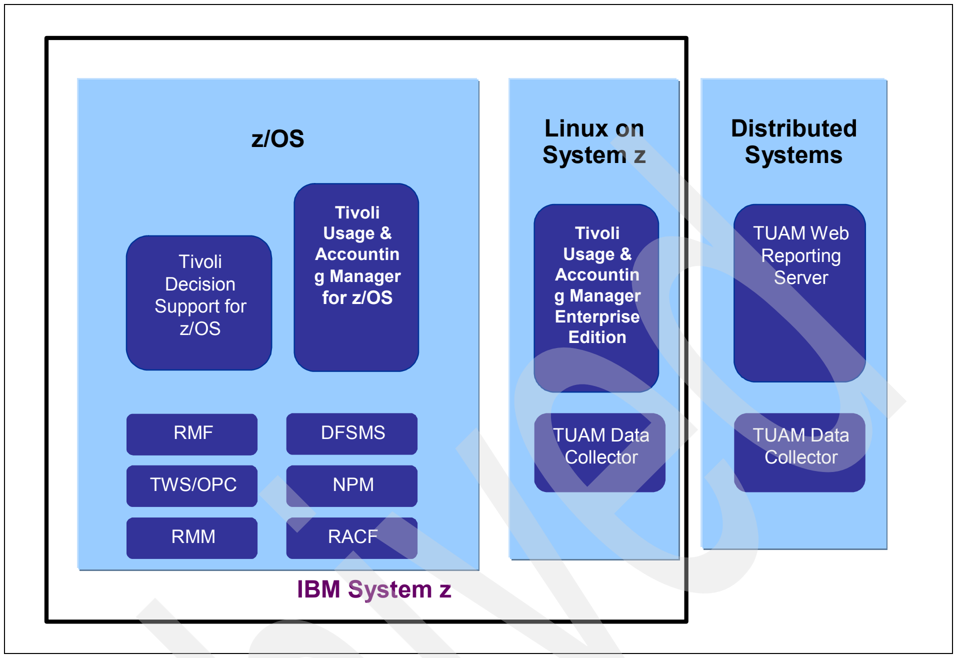

z/OS
Tivoli Decision Support for z/OS
Tivoli Usage & Accountin g Manager for z/OS
RMF
DFSMS
TWS/OPC
NPM
RMM
RACF
IBM System z
Linux on System z
Tivoli Usage & Accountin g Manager Enterprise Edition
TUAM Data Collector
Distributed Systems
TUAM Web Reporting Server
TUAM Data Collector

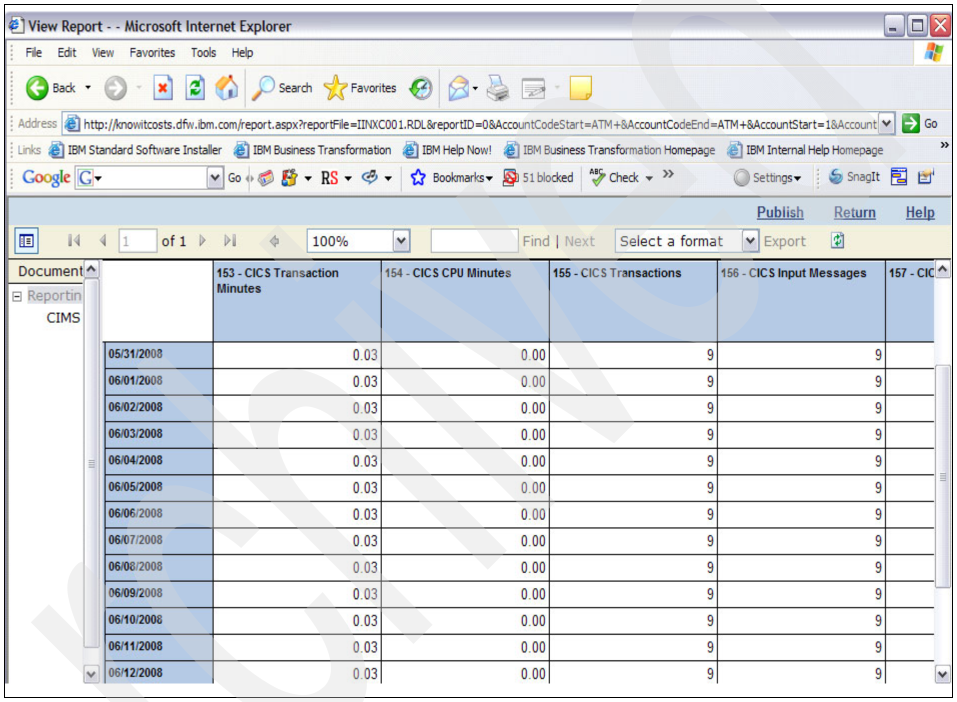
Figure 7-52 Accounting management solution

IBM Tivoli Usage and Accounting Manager (TUAM)

The *IBM Tivoli Usage and Accounting Manager (TUAM)* software family measures, collects, analyzes, reports, and, optionally, bills the utilization and costs of different computing resources including applications, servers, storage, networks, databases, messaging, and many other shared services. TUAM consolidates all types of usage metering records into a common and integrated reporting structure with its rich Data Collectors associated with operating systems, databases, Internet infrastructure, network and printing, and customized usage data import collection from any application or system. TUAM then generates reports, invoices, and summarized files showing resource consumption and monetary expenditures for each functional unit within the organization.

The TUAM Data Collector collects information and allocates costs to the appropriate user and provides support for a wide range of z/OS and System z sources. TUAM also provides cross-platform detailed reports and invoicing and enables measurement of resource usage and effective charge back. TUAM creates customizable Web-based reports with an instant drill-down capability and lets you invoice users based on popular cost allocation methods. TUAM allows alignment of IT costs with company priorities and has the ability to account for individual department's use of key applications, servers, and other IT resources by providing an extremely flexible end-to-end tool that helps you improve IT cost management. You can track, allocate, and invoice based on actual resource use by department, user and many additional criteria.

TUAM Enterprise Edition collects, analyzes, and bills based on usage and costs of shared distributed systems including Linux on System z. TUAM for z/OS is an option of Tivoli Decision Support (TDS) for z/OS. Figure 7-53 displays a sample TUAM CICS report.


View Report - - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links IBM Standard Software Installer IBM Business Transformation IBM Help Now! IBM Business Transformation Homepage IBM Internal Help Homepage
Google Go RS Bookmarks 51 blocked Check Settings SnagIt
Publish Return Help
1 of 1 100% Find | Next Select a format Export
Document
Reportin
CIMS

http://knowitcosts.dfw.ibm.com/report.aspx?reportFile=IINXC001.RDL&reportID=0&AccountCodeStart=ATM+&AccountCodeEnd=ATM+&AccountStart=1&Account

	153 - CICS Transaction Minutes	154 - CICS CPU Minutes	155 - CICS Transactions	156 - CICS Input Messages	157 - CIC
05/31/2008	0.03	0.00	9	9	
06/01/2008	0.03	0.00	9	9	
06/02/2008	0.03	0.00	9	9	
06/03/2008	0.03	0.00	9	9	
06/04/2008	0.03	0.00	9	9	
06/05/2008	0.03	0.00	9	9	
06/06/2008	0.03	0.00	9	9	
06/07/2008	0.03	0.00	9	9	
06/08/2008	0.03	0.00	9	9	
06/09/2008	0.03	0.00	9	9	
06/10/2008	0.03	0.00	9	9	
06/11/2008	0.03	0.00	9	9	
06/12/2008	0.03	0.00	9	9	

Figure 7-53 Tivoli Usage & Accounting Manager - CICS report by User ID

IBM Tivoli Decision Support (TDS) for z/OS

IBM Tivoli Decision Support (TDS) for z/OS measures and reports performance and accounting data for host and distributed systems. TDS for z/OS collects systems management data from the log data of various programs, processes it, and stores it in a database. TDS for z/OS integrates with other Tivoli and IBM systems management products such as RMF, DFSMS, Tivoli Workload Scheduler (TWS), NPM, NetView for z/OS, RACF, RMM, Linux on System z, WebSphere Application Server for z/OS, and so forth. For example, with TDS for z/OS, we can track the IT dollars from start to finish with the already mentioned Tivoli Usage and Accounting Manager for z/OS (TUAMz) option. TDS for z/OS consists of the base product and several individual feature packs, such as the System Performance feature, CICS performance feature, IMS performance feature, and Network performance feature, together with its modern Web browser interface for report generation and publishing. TDS for z/OS helps in maintaining centralized control and performance-reporting efficiency.

7.7.3 Asset management solution

There is a growing need for identifying, tracking, and managing software licenses and hardware in today's already complex IT environment. A solid software asset management solution is imperative for helping to control unauthorized purchases and mitigating the legal and financial risks of noncompliant software installations. Large, heterogeneous IT organizations often have multiple asset repositories.

The integrated solution enables us to easily link to enterprise business applications as well as auto discovery and systems management applications, so that asset information can be maintained and viewed through a single interface. This in turn enables you to quickly and efficiently manage the IT investments and better control burgeoning costs related to compliance, procurement, budgeting and forecasting.

Figure 7-54 shows a high level view of the Asset management solution.

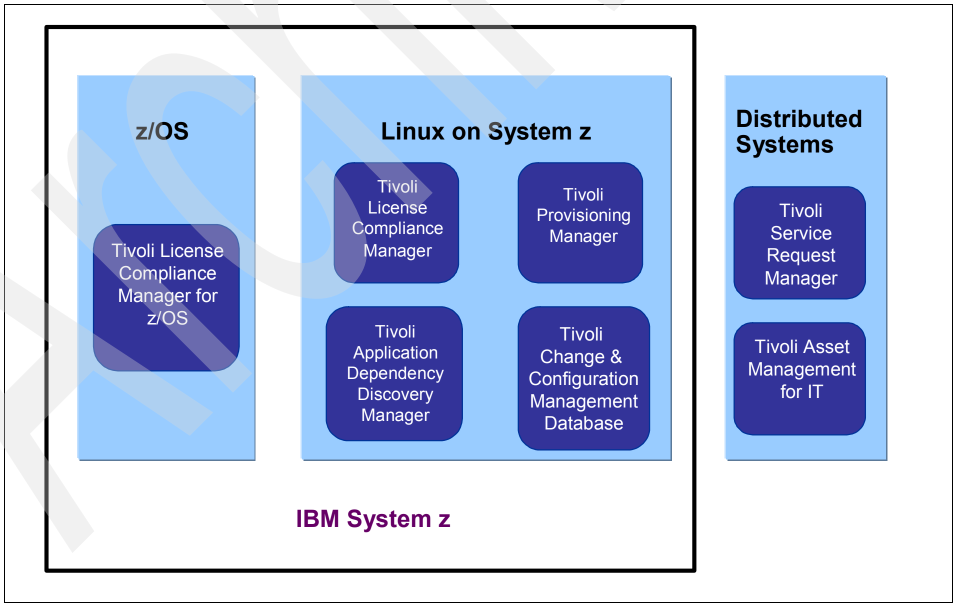

z/OS
Tivoli License Compliance Manager for z/OS
Linux on System z
Tivoli License Compliance Manager
Tivoli Provisioning Manager
Tivoli Application Dependency Discovery Manager
Tivoli Change & Configuration Management Database
Distributed Systems
Tivoli Service Request Manager
Tivoli Asset Management for IT
IBM System z

Figure 7-54 Asset management solution

IBM Tivoli Asset Management for IT

The *IBM Tivoli Asset Management for IT* portfolio provides the tools and solutions to manage hardware and software assets through their entire life cycle from planning for acquisition through retirement. IBM Tivoli Asset Management for IT enables efficient and effective management of the IT asset life cycle to lower cost, mitigate license and regulatory compliance risk, and better align IT with business goals. It provides features such as contract management and procurement, interactive action-based workflows, and reconciliation of deployed versus authorized assets. A query wizard allows users to define and schedule their own reports.

IBM Tivoli Asset Management for IT is part of the IBM Asset Management product offerings; we do not discuss all of them here.

Other asset management products

Some other products in the solution suite that help in achieving greater efficiency in asset management are the following:

- IBM Tivoli Service Request Manager
  - Allows customers to streamline service desk operations and reduce costs by unifying key service support and asset management processes.
- IBM Tivoli Change and Configuration Management Database (CCMDB)
  - Discovers and federates IT information spread across the enterprise, including details about servers, storage devices, networks, middleware, applications, and data.
  - CCMDB requires IBM Tivoli Application Dependency Discovery Manager (TADDM).
- IBM Tivoli License Compliance Manager for distributed systems (including Linux on System z)
  - Identifies distributed software inventory, measures use activity, and links complex licenses to help manage costs and compliance.
- IBM Tivoli License Compliance Manager for z/OS
  - Automatically detects software running within your environment and related use patterns.
  - Identifies z/OS software inventory, measures use activity, and links complex licenses to help manage costs and compliance.
  - Reports on the MIPS capacity of each LPAR under which software is running.
  - Improves Disaster Recovery planning with software use activity information.

– Identifies software inventory with no or low use activity to reduce unnecessary license fees.

IBM Tivoli License Compliance Manager for z/OS and IBM Tivoli License Compliance Manager for distributed systems provide a robust end-to-end software asset management solution.

Figure 7-55 shows a high level view of the Tivoli License Compliance Manager for z/OS architecture.

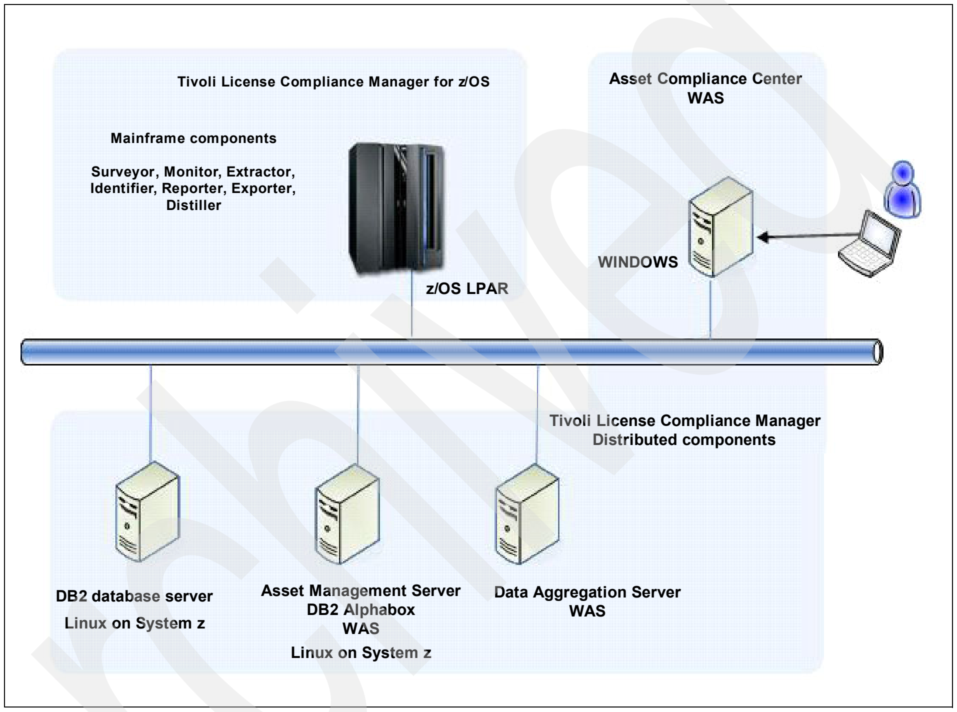

Tivoli License Compliance Manager for z/OS
Mainframe components
Surveyor, Monitor, Extractor, Identifier, Reporter, Exporter, Distiller
z/OS LPAR
Asset Compliance Center WAS
WINDOWS
Tivoli License Compliance Manager Distributed components
DB2 database server
Linux on System z
Asset Management Server DB2 Alphabox WAS
Linux on System z
Data Aggregation Server WAS

Figure 7-55 Tivoli License Compliance Manager for z/OS architecture

- IBM Tivoli Provisioning Manager
  – Built on SOA.
  – A portfolio of products that enables automation of complex tasks ranging from bare-metal OS provisioning to software deployment, to desired-state maintenance and policy-based orchestration.
  – Able to execute changes, configuration, and maintenance of servers and virtual servers, operating systems, middleware, applications, storage, and network devices through a graphical user interface.
  – Provides compliance reporting and remediation.

Figure 7-56 shows a high level view of the Tivoli Provisioning Manager architecture.

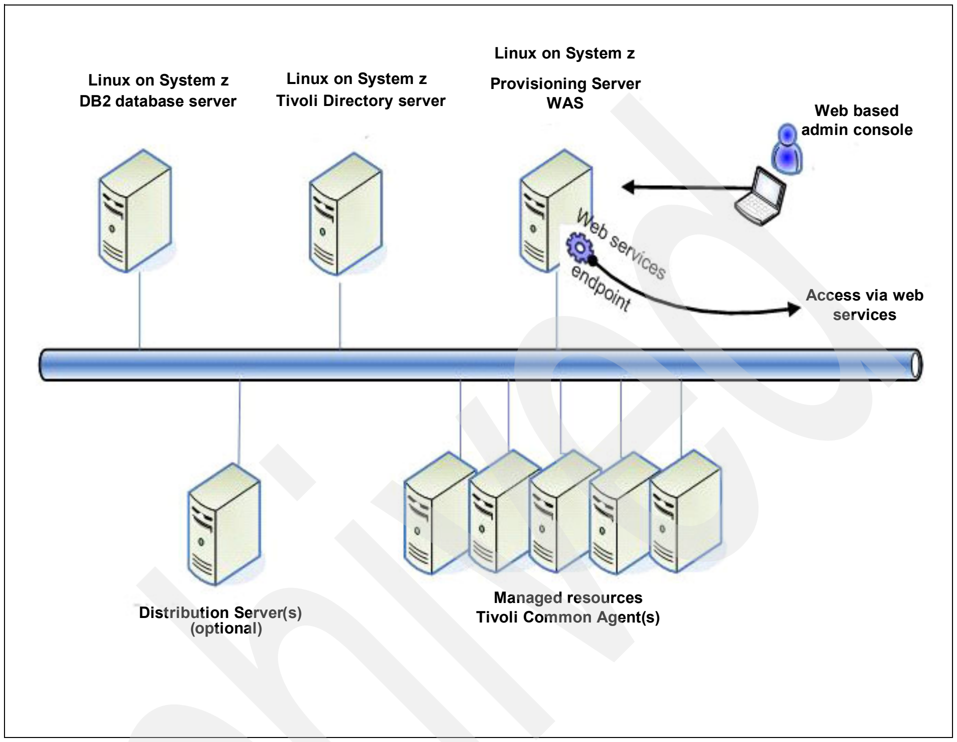


Figure 7-56 Tivoli Provisioning Manager architecture

7.8 Storage management

There is a vast amount of data within today's enterprise and it is growing every second. Storage management ensures that data is maintained in a controlled and effective manner until it is disposed of. Data has a life cycle, which based on the type of data, as well as the standards and regulations associated with that data. To provide effective management of data, a portfolio of managed data is maintained by the IT organization. This helps determine where the data is, as well as life cycle practices to be performed. This process performs all backups and restores of data. Once data is no longer needed by the enterprise, it can be disposed of.

The purpose of the storage management process is to ensure that all data necessary in providing and supporting business and operational services is available for use and is actively managed from creation and introduction until final disposal or destruction.

7.8.1 Key architectural decisions with respect to Storage management

Consider the following questions when designing a storage management solution:

- What are the requirements for the storage management processes?
- Do the existing data backup and restore processes and strategies effectively protect the business interest?
- Do accurate and appropriate storage management tools exist?
- What storage management model is appropriate for the organization? Centralized or de-centralized?

7.8.2 Storage management solution

The storage management solution includes a process to facilitate the identification and cleanup of files in the environment that do not belong on primary or tier one storage. It further provides the capability for customers to define Service Level Objectives (SLOs) and capabilities for data and servers in the environment to facilitate tiered storage management. These policies will be presented when companies are performing change, configuration, release, problem, incident, and service level management for data and storage in their environment. The solution will get an enterprise view of storage capacity and utilization and performance to optimize use and availability. It gets better control of storage management costs by more effectively leveraging capacity and tiering of information storage resources. It also automates data protection and retention policies, and storage network and resource management. Figure 7-57 on page 451 shows the storage management solution using IBM Tivoli products.

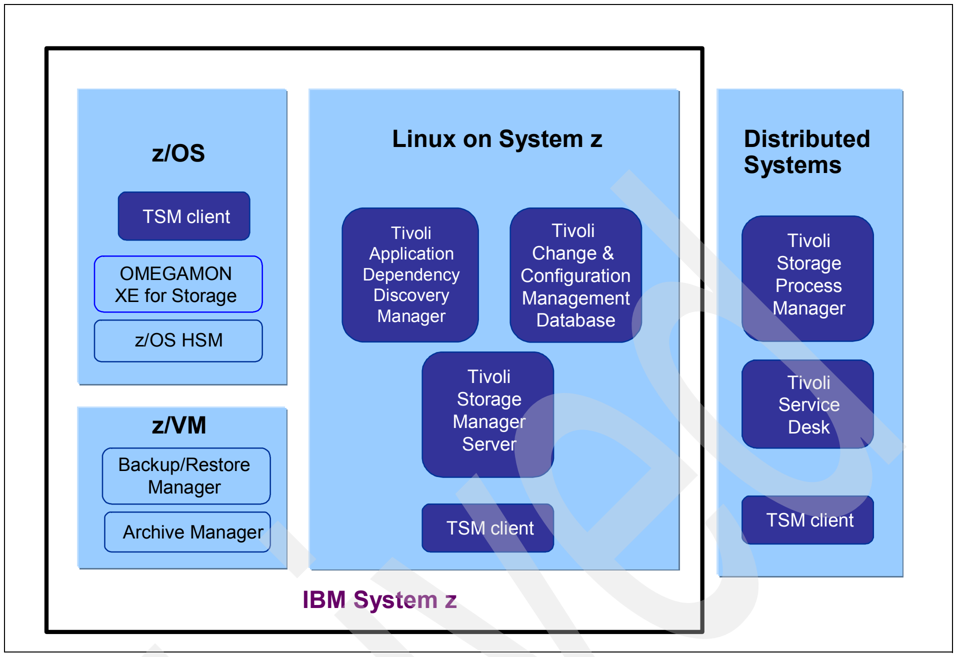


Figure 7-57 Storage management solution

IBM Tivoli Storage Process Manager (ITSPM)

IBM Tivoli Storage Process Manager (ITSPM) aligns with ITIL best practices by implementing change, configuration, and release management around storage management. It enhances the current change, configuration, and release management by extending them with storage-related tasks and process steps.

ITSPM takes a holistic view of the IT storage environment to facilitate and align storage management to the business objectives of an organization. As a productivity tool, the IBM Tivoli Storage Process Manager manages and coordinates simple and complex storage provisioning, configuration, and incident management tasks. This Portal-based solution helps customers implement a storage management process. It coordinates and manages storage change, configuration, and incident management throughout the life cycle using an easy-to-use interface. Storage provisioning and configuration changes can be automated by connecting the implementation activity directly to the operational management product, such as IBM Tivoli Storage Manager, to perform the operational task, monitor status of the task, and verify completion of the task. Storage provisioning and configuration changes can be planned and implemented on a predefined schedule or can be ad hoc to support unplanned emergency fixes and updates. ITSPM creates and registers ITIL-based configuration items (CIs) for storage objects such as devices, switches, and

SANs, helps automate management policies, and provides a fully defined model of storage attributes and relationships in the IBM Tivoli Change and Configuration Management Database (CCMDB).

Note: Maximo® Version 6.2.1 must be installed with IBM Maximo Change Manager for IBM Change and Configuration Management Database 6.2.1 and IBM Tivoli Service Desk 6.2.1 enabled before you can install IBM Tivoli Storage Process Manager.

IBM Tivoli Change and Configuration Management Database (CCMDB)

IBM Tivoli Change and Configuration Management Database (CCMDB) discovers and federates IT information spread across the enterprise, including details about servers, storage devices, networks, middleware, applications and data. We have already discussed this component in previous sections and therefore do not discuss it further here. There are two *Discovery Library Adapters (DLAs)* that can be used to discover z/OS resources and populate CCMDB:

- z/OS DLA

  This has no special pre-requisites and does not require a live OMEGAMON environment or live TBSM environment. Simply submit the batch job and discovery will take less than 1 minute.

- Tivoli Monitoring Services (TMS) DLA

  For sites that have OMEGAMON XE deployed, the TMS DLA can be used to discover OMEGAMON XE agents and map them to z/OS resources. Enables TADDM to provide context-sensitive launch to TEP and query TEPS for the status.

See Figure 7-44 on page 428 for an illustration of CCMDB and TADDM architecture.

IBM Tivoli Storage Manager (ITSM)

The *IBM Tivoli Storage Manager (ITSM)* family of products delivers a single point of control and administration for your storage management needs, helping to increase the efficiency of your IT operations and control costs related to storage management. It performs a variety of tasks through policy-based automation, including:

- Backup and recovery
- Archiving and retrieval
- Disaster recovery

- Space management
- On-line database and application protection
- Bare-machine recovery
- Continuous data protection

ITSM provides centralized, Web-based administration and intelligent data move-and-store techniques to ease storage management.

In conjunction with the IBM Tivoli Systems Automation for Multiplatforms (TSA), which provides a highly available cluster environment for ITSM infrastructures, it offers mainframe-like high availability by using fast detection of outages and sophisticated knowledge about application components and their relationships. It provides quick and consistent recovery of failed resources and whole applications either in place or on another system in the cluster without operator intervention.

z/OS HSM

z/OS HSM (DFSMShsm) is a licensed program on z/OS that automatically performs space management and availability management in a storage device hierarchy. DFSMShsm makes sure that space is available on the Direct Access Storage Device (DASD) volumes so that we can extend the old data sets and allocate new ones. DFSMShsm also makes sure that backup copies of the data sets are always available if the working copies are lost or corrupted. It has no direct relationship with ITSM but it is one of the well recognized data management tools on z/OS.

Integrated Removable Media Manager (IRMM)

Integrated Removable Media Manager (IRMM) is a new robust systems management product for Linux on System z that manages open system media in heterogeneous distributed environments and virtualizes physical tape libraries, thus combining the capacity of multiple heterogeneous libraries into a single reservoir of tape storage that can be managed from a central point. IRMM is designed to provide centralized media and device management, dynamic resource sharing, and enhanced media protection and security. IRMM extends IBM's virtualization strategy to tape library resources (drives and cartridge pools). IRMM interfaces with DFSMSrmm (RMM) to provide enterprise-wide tape management. See Figure 7-58 on page 454.

With the integration with TEP it provides a single interface for storage management, for example, event alerts. It leverages the information from multiple sources, and enhances and improves problem resolution and automation capabilities.

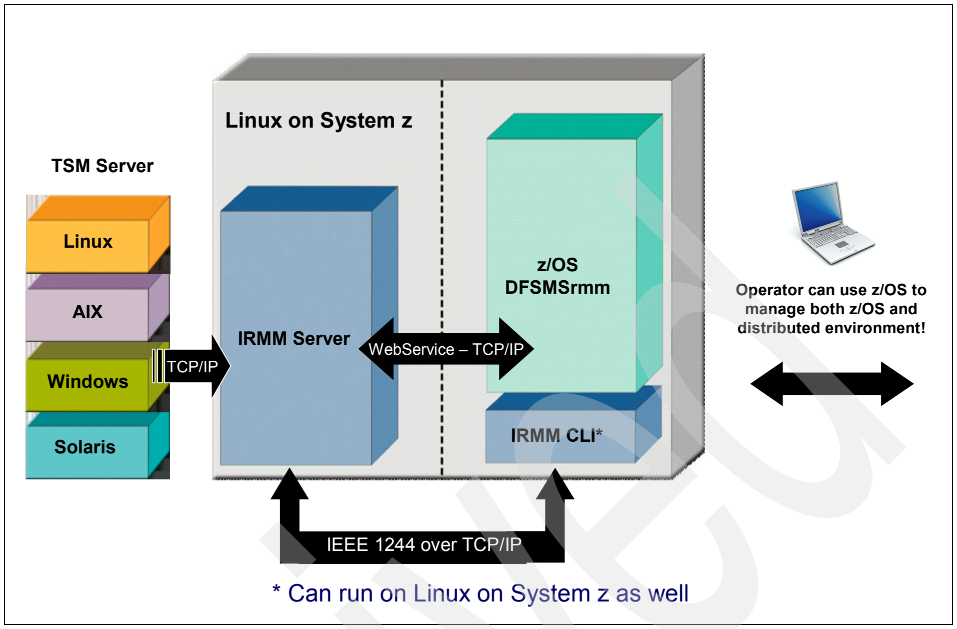


Figure 7-58 Tape management solution

In conjunction with other Tivoli Total Storage management solutions such as OMEGAMON XE for Storage on z/OS, Advance Reporting for DFSMShsm, Advanced Audit for DFSMShsm, Advanced Catalog Management for z/OS, Allocation Optimizer for z/OS, Backup/Restore Manager for z/VM, and Archive Manager for z/VM, a complete System z Total Storage management solution can be achieved.

7.9 Centralized Service Management using System z

In the previous sections, we have discussed several major management functions in IT Services Management separately and suggested different solutions with respect to SOA. IBM System z now has the capabilities including process automation, new and mixed workloads, SOA hub, security hub, data warehousing hub, and so forth. System z growth transformation requires a new approach. Therefore, there is a requirement for transformation from legacy platform management to managing System z as a hub for mixed enterprise workloads, server virtualization, and end-to-end services. We need an integrated solution that enables System z as a highly reliable hub for efficient management of business and IT services that span the enterprise.

In this section, we introduce and describe an integrated solution that uses IBM System z as the core platform for managing services that span heterogeneous operating systems and platforms. This solution integrates various sets of IBM software solutions designed to substantially improve management of mainframe environments. These solutions also help organizations deploy Service Management strategies regardless of their platforms and processes. It will benefit companies who want to exploit the operational advantages of System z to deliver and expand enterprise services managed as a utility.

An integrated management solution using System z as a hub provides the following advantages:

- A compelling new way to leverage and expand the value of the System z platform, providing a concrete set of integrated solutions and process automation focused on managing business and IT services across the enterprise.
- Enhanced integration of the System z platform to help transform to a service-centric management model.
- Provides the required visibility, control, and automation across the service, technical support, operations, security, and financial domains and enable the transformation from a System z resource management focus to a centralized System z Service Management model.
- Demonstrates how “the whole is greater than the sum of its individual parts.”

IBM Tivoli Service Management Center for System z is a new, integrated set of IBM software solutions designed to substantially improve management of mainframe environments.

The core set of IBM Tivoli Service Management Center for System z solutions provides:

- Service automation within System z.
  Extends System z operations automation with process automation and Service Management.
- Centralized service automation solution across all systems.
  Exploits System z advantages by driving business service and ITIL process management across the enterprise.
- IBM Service Management Center for System z enables organizations to strategically use their mainframe as an integrated, enterprise-wide hub for efficiently managing business and IT services.
- Provides an integrated approach focused on ensuring that IT has the required visibility, control, and automation in place to efficiently and effectively deliver business services.

- Provides easy to use and integrated monitoring tools for end-to-end visibility of the enterprise and also proactively warn you about critical situations, greatly simplifying management.
- Provides integrated tools for end-to-end control to help you effectively schedule and execute planned changes, optimize asset utilization, and understand the cost of service delivery.
- Tight and easy integration across solutions makes it easier to automate operations across applications and environments, thereby making the business process simple, efficient, and effective.

Figure 7-59 shows the IBM Tivoli Service Management Center for System z solution areas and the best practices.

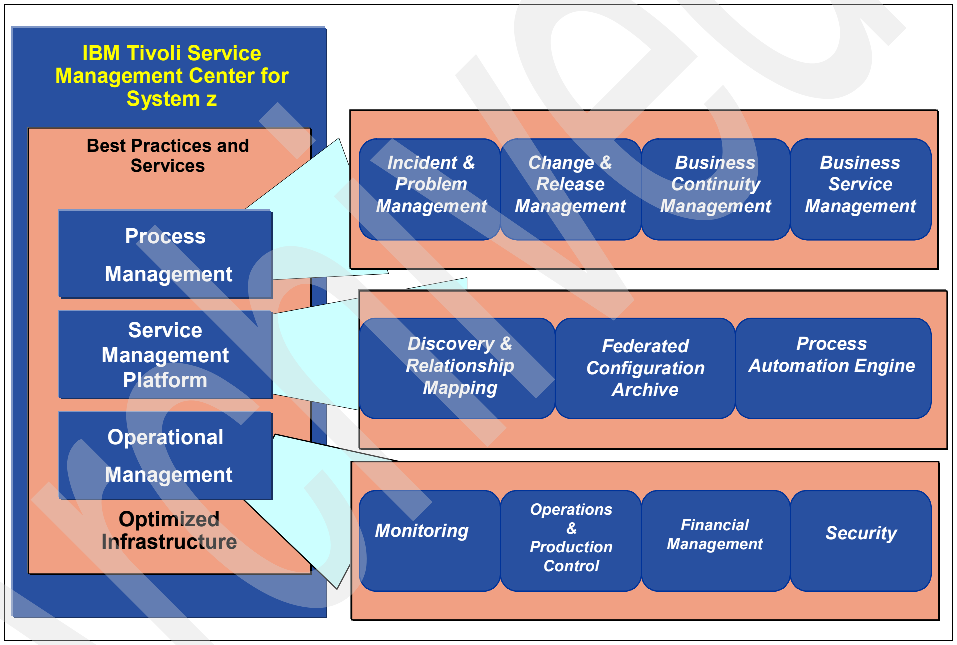

IBM Tivoli Service Management Center for System z
Best Practices and Services
Process Management
Service Management Platform
Operational Management
Optimized Infrastructure
Incident & Problem Management
Change & Release Management
Business Continuity Management
Business Service Management
Discovery & Relationship Mapping
Federated Configuration Archive
Process Automation Engine
Monitoring
Operations & Production Control
Financial Management
Security

Figure 7-59 IBM Tivoli Service Management Center for System z solution areas and the best practices

The IBM Tivoli Service Management Center for System z is not a product, but rather a portfolio of integrated solutions, organized into management domains, offered through soft bundles of solutions. It is a new and unique set of integrated solutions that substantially improves the management of mainframe environments and enables you to host enterprise-wide Service Management processes and capabilities on the mainframe. By hosting your critical Service Management solutions on System z, you can leverage this highly reliable, available, and secure platform. IBM Tivoli Service Management Center for

System z can be implemented in orderly stages. IBM Tivoli Service Management Center for System z offers a comprehensive set of solutions to support the organizations unique business and IT objectives. It can be started with any combination of these integrated solutions.

Figure 7-60 shows the various entry points for the IBM Tivoli Service Management Center for System z.

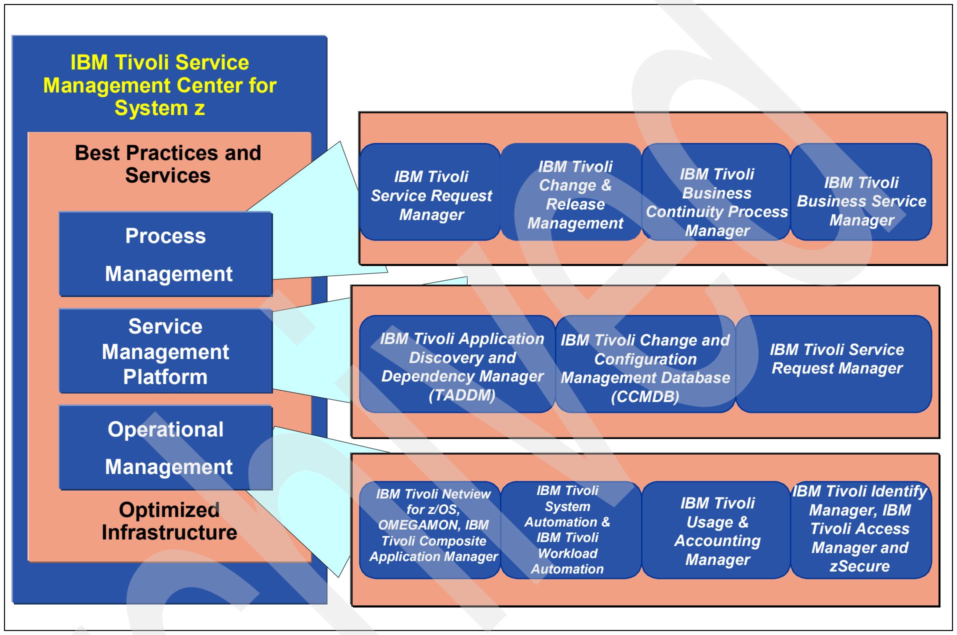

IBM Tivoli Service Management Center for System z
Best Practices and Services
Process Management
Service Management Platform
Operational Management
Optimized Infrastructure
IBM Tivoli Service Request Manager
IBM Tivoli Change & Release Management
IBM Tivoli Business Continuity Process Manager
IBM Tivoli Business Service Manager
IBM Tivoli Application Discovery and Dependency Manager (TADDM)
IBM Tivoli Change and Configuration Management Database (CCMDB)
IBM Tivoli Service Request Manager
IBM Tivoli Netview for z/OS, OMEGAMON, IBM Tivoli Composite Application Manager
IBM Tivoli System Automation & IBM Tivoli Workload Automation
IBM Tivoli Usage & Accounting Manager
IBM Tivoli Identify Manager, IBM Tivoli Access Manager and zSecure

Figure 7-60 IBM Tivoli Service Management Center for System z - products

The IBM Tivoli Service Management Center for System z solution consists of the following major solutions. As you can see, most of them were already discussed in the previous sections of this chapter:

- IBM Tivoli OMEGAMON XE on z/OS
- IBM Tivoli Composite Application Manager (ITCAM)
- IBM Tivoli System Automation (TSA)
- IBM Tivoli NetView for z/OS
- IBM Tivoli Workload Automation
- IBM Tivoli Usage & Accounting Manager (TUAM)
- IBM Tivoli Identity Manager (TIM)
- IBM Tivoli zSecure

- IBM Tivoli Business Service Manager (TBSM)
- IBM Tivoli Application Discovery and Dependency Manager (TADD)
- IBM Tivoli Service Request Manager
- IBM Tivoli Change and Configuration Management Database (CCMDB)

The IBM Tivoli Service Management Center for System z solution with fully integrated products facilitates ITIL alignment and the fully integrated product architecture with process automation and tooling is designed around ITIL.

8

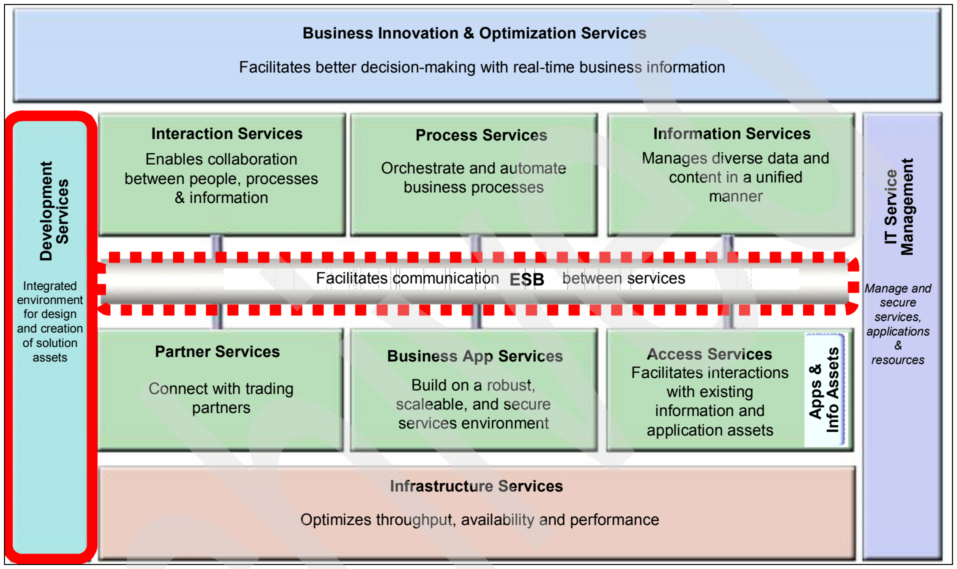
Service life cycle management

In this chapter we focus on the left vertical box of the IBM SOA Reference Architecture, called *Development services*. However, the subject is bigger than just providing tools for development of services. Tools, processes, procedures, and an organization need to be in place to continuously develop and maintain services. The traditional development and maintenance approach is significantly different than that required in an SOA model. We now need to design, develop, test, and deploy at the service level rather than at the application release level. Services might not belong to a specific application, but be shared by many different applications. A service needs to have an owner, responsible for its life cycle.

In this chapter we refer to this area as *service life cycle management*.

8.1 Definition, scope, and functions

In this section we define the scope and purpose of service life cycle management. Figure 8-1 locates this discipline within the IBM SOA Reference Architecture. As shown in the figure, this area of concern includes not only Development Services, but also extends into the Enterprise Service Bus (ESB) layer by means of service registry solutions.



Business Innovation & Optimization Services
Facilitates better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enables collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manages diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
Facilitates communication ESB between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitates interactions with existing information and application assets
Apps & Info Assets
Infrastructure Services
Optimizes throughput, availability and performance

Figure 8-1 Services life cycle management in the IBM SOA Reference Architecture

8.1.1 Definition

Service life cycle management prescribes how services are defined, developed, deployed, and managed.

SOA and the services it delivers provide increased flexibility that cuts across the boundaries of traditional business unit silos. The cross-organization nature of SOA therefore makes service life cycle management vital to any organization.

Figure 8-2 shows the processes you would typically see during a service's life cycle as defined in *Service Oriented Modeling and Architecture (SOMA).*

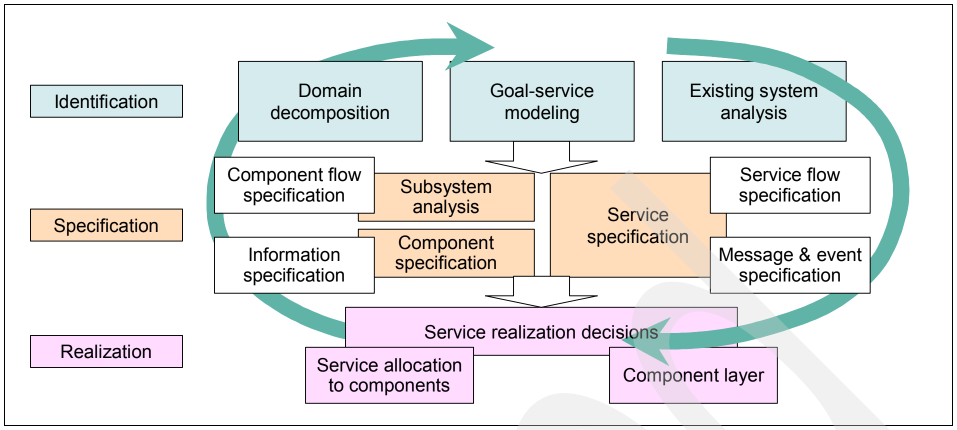


Figure 8-2 Process and stages of a service as defined in SOMA

8.1.2 Scope

With specific reference to System z, we look at service definition and development from a top down as well as a bottom up approach. We also look at System z specific tooling to assist the life cycle management.

8.1.3 Function

Service life cycle management defines and provides processes that need to be implemented and governed to ensure that the benefits offered by SOA, particularly reuse and responsiveness, are realized.

8.2 SOA governance

SOA governance is a specialization of governance within IT.

Ideally, SOA governance should enable a decision structure that can enforce certain behavior both within and between business units and IT.

This behavior is necessary due to the very fragmented nature of components that are composed to form a business process. The behavior defines roles within business and IT that participate in these decision structures. It is therefore probable that your current organizational structure might not map to these new roles.

The processes dealing with service identification, design, implementation, and deployment all form part of service life cycle management and are comparable to those in more traditional development efforts. However, SOA governance also deals with such issues as:

- **Ownership**
  Services bisect traditional business unit silos. Responsibility for SOA components therefore typically does not map well to traditional business and project structures. This responsibility stretches from initial funding to monitoring.
- **Discovery**
  The ability to find reusable components to form part of business processes is key to realizing returns from an investment in SOA.
- **Versioning**
  The ability to support multiple version of a service in parallel.
- **Deprecation**
  Older versions of services reach a stage of deprecation. Existing consumers must be identified and change coordinated.
- **Monitoring**
  You need to monitor whether services comply with the agreed SLAs.

As mentioned, SOA governance is a super-set of service life cycle management. Although this chapter deals with service life cycle management, we feel SOA governance and its close coupling with service life cycle management warrants careful attention.

It is vital that SOA governance be in place even during the early stages of an SOA implementation. Any organization might be able to manage components during initial stages, but manageability quickly is lost as services and their consumers proliferate.

8.2.1 Enterprise architecture framework

IT organizations should standardize on a framework that defines architecture standards, development processes, design patterns, and tools. Most IT organizations will already have adopted one of the standard application life cycle management processes and modified it to fit its own needs. In addition, there are other well known architecture frameworks such as Zachman, *Federal Enterprise Architecture (FEA)* and *The Open-Group Architecture Framework (TOGAF)*. IT must adopt an architecture framework enterprise-wide to be successful in implementing SOA.

8.3 The life cycle of a service

The life cycle management philosophy behind services in an SOA is different than with traditional application life cycle management. That is why we have chosen not to use the term “application life cycle management” anymore in an SOA context. In an SOA we manage services and not applications. A service is a loosely-coupled atomic component not only from a design and coding perspective, but from a life cycle management perspective as well. Certain services will need to change at given times, triggered by various events. Examples of events that could trigger updates of a service are:

- One specific business requirement comes along, resulting in the change of a specific business rule. This can be, for example, a change in the discount logic on certain insurance products. This change could affect one or maybe a few services. Be aware, though, that it could affect hundreds of consumers of the affected services!
- A merger or acquisition leads to the merging of different business rules or the obsoleting of certain business rules. This could lead to updates and deletions of one or many services.
- A business unit has decides it needs a new application with new functions. This could lead to the addition of new services, but also changes in existing ones to accommodate the new application. Also, new services might be created out of existing legacy code. This is where the discovery comes into play.

In any case, our focus is now on the service level and we have to decide for each service what to do next. Design, development, testing, and deployment occur per service and these services are fairly independent of one another.

Figure 8-3 shows the major choices in service life cycle management.

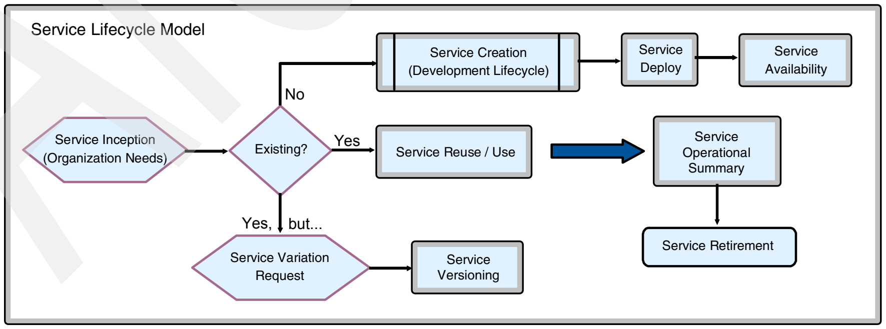


Figure 8-3 The continuous life cycle of s service

Figure 8-4 illustrates that services are shared between business units, but managed only in one place.

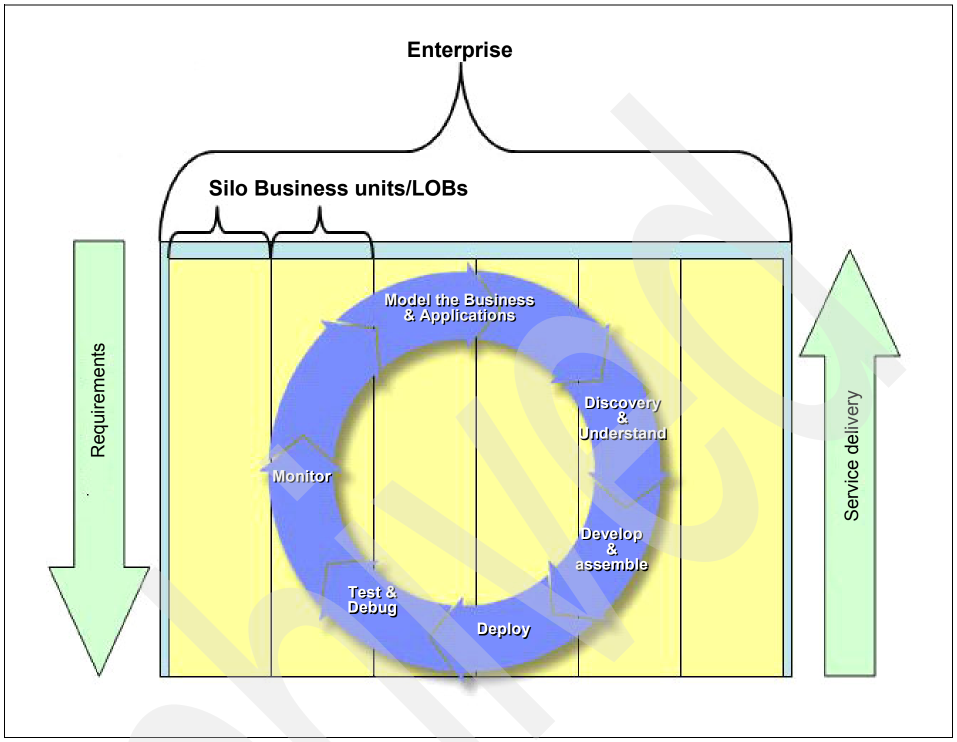


Figure 8-4 Services shared between business units

8.4 Service identification

One of the most crucial and challenging steps in service life cycle management is identifying services. There are three approaches that can be followed:

- Top down

  A *top down* approach is one that is driven from a business perspective. This is often seen as a strategic approach to SOA. The initiative and funding in such situations is often project specific. Very few organizations are willing to risk the investment or disruption of wide ranging, large scale adoption of an SOA. This specific project should realize well documented business requirements. Services can then be realized from what is typically use cases or some other form of business process documentation. Domain decomposition forms the part of this process whereby the business domain is decomposed into business functions and processes. High level use cases can then be derived, and this will form input to the service identification. *Process orchestration* is then used to provide an atomic solution to the business process.

- Bottom up

  The *bottom up* approach is driven from an IT perspective. Typically, existing applications are analyzed and refactored into usable functions. This can be seen as a de-coupling of functions from an existing application. In monolithic legacy applications it might be necessary to refactor the identified functions into individual modules. A common error with this approach is that the services realized are not business focused — they provide solutions to IT problems. This common mistake can be prevented by exercising strict service governance in the form of a service review process.

  There are benefits to this approach, however. A bottom up approach can be useful to provide some deliverables to an organization during the initial stages of implementing an SOA. Once roles and processes have been defined to support governance, and the business units have come to understand and embrace an SOA model, it is advisable to switch to a top down approach.

  There is a second benefit to this approach. In a bottom up approach the *project view only* of a service is less likely then in a top down approach. A project view of a service is a situation whereby a service is realized without understanding its usefulness to the enterprise beyond the scope of this project. This obviously affects service reuse, one of the key benefits of SOA.

- Middle out

  The middle out approach consists of *goal-service modeling* to validate and unearth other services not captured by either top down or bottom up service identification approaches. It ties services to goals and sub-goals, key performance indicators, and metrics.

Once the identification process is complete the different approaches merge into one stream and the service continues on its life cycle.

8.4.1 Top down service identification

Domain decomposition describes the identification of processes within key functional areas (business domains). The IBM *Service Oriented Modeling and Architecture (SOMA)* method allows for the use cases and business processes to identify candidate services. These candidates should then be subjected to a litmus test to ensure such things as their strategic value; granularity level is another such governance issue. Probably the most important governance issue is “Does this service already exist?” If this governance is not in place, services will proliferate and the benefits of SOA will not be realized. The service can now be specified according to your particular model.

Assuming the service has passed scrutiny during the governance process the modeling of a service specification can now take place. The service specification should provide all details of a service that a potential consumer might require. The potential consumer will be matching a requirement for business functionality to a service specification. The consumer therefore requires such information as:

- Service name, which should at least provide a purpose and domain
- Service request and response
- High level function

It is important to remember that this service specification does not detail the processing involved in achieving the stated function (meaning its actual implementation). Service specifications should typically not have to describe processing. If there is a tendency towards having to describe processing you will probably find the service is too course grained. This might be due to process choreography being embedded within a single service. If this was a conscious decision then it might be acceptable, but it should be the exception and not the rule.

It is also important to remember that your model for service specification should provide enough detail to potential consumers for them to make an informed decision as to the suitability of a service, meaning, will this service meet the business goals stated in the use case or specification? Ideally the consumer should not have to be referred to various free format text documents to establish a service's suitability.

Figure 8-5 on page 467 illustrates this process of domain decomposition.

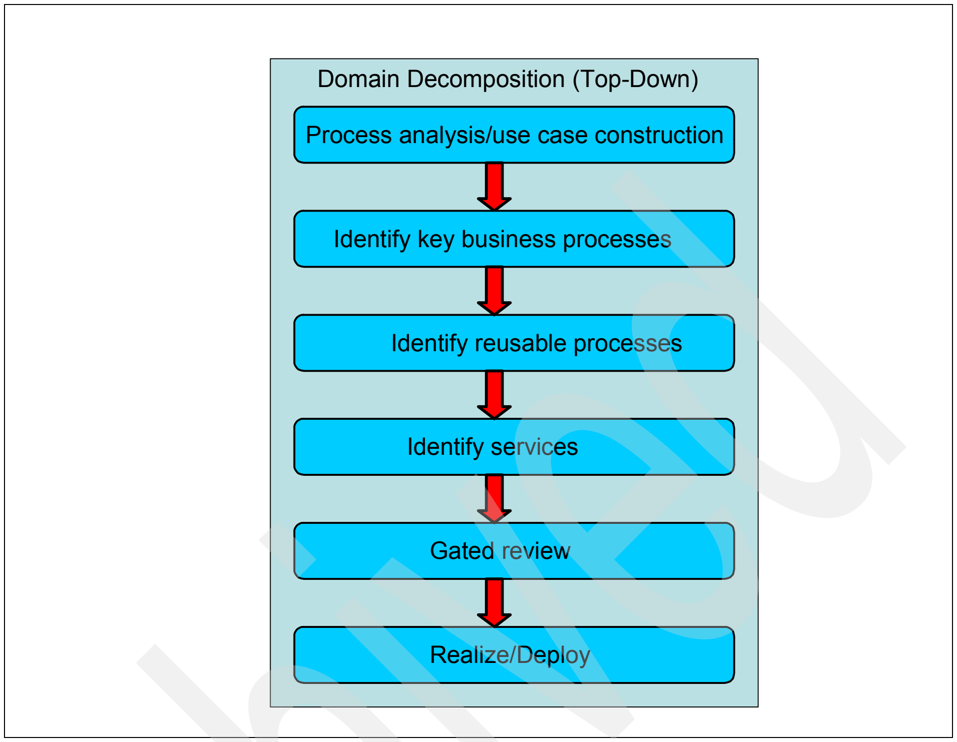

Domain Decomposition (Top-Down)
Process analysis/use case construction
Identify key business processes
Identify reusable processes
Identify services
Gated review
Realize/Deploy

Figure 8-5 Domain decomposition

Process and service modeling

WebSphere Integration Developer (WID) assists in transforming identified business processes using WebSphere Business Modeler into a *Business Process Execution Language (BPEL)* representation. This BPEL-defined process can later be run and managed using a runtime engine such as the WebSphere Process Server (WPS).

Since this chapter concentrates on the underpinnings for core system service enablement, we do not examine those tools in detail here. Tools such as the WebSphere Business Modeler are used primarily in top down application designs, where the process or composite application is defined first, and this design drives the underlying service granularity. In this section, we are looking at bottom up service enablement, where existing applications are refactored into SOA-based components.

8.4.2 Bottom up service identification

The bottom up approach to service identification is driven by analysis of existing applications. In this process applications are analyzed and functionality is refactored. This refactoring often takes the form of modularizing a previously monolithic application. The resulting functionality can then be used to form a service. Some basic choreography might be necessary to provide the correct level of granularity. Too course grained and you reduce flexibility and therefore the possibility of reuse; too fine grained and you have chatty services that will probably perform poorly. Once a candidate service has been identified it must follow the same governance process as described in the previous discussion about a top down approach. This governance or gated review process is, in fact, even more necessary than for a service identified via a a top down process. The litmus test for a service produced through this mechanism is really "Is it likely to be of use as part of a business process, and if so, how often?" This is often seen by many as the downfall of a bottom up approach. The fact that services are identified without a business requirement driving the process requires a thorough governance process to ensure services identified are likely to add value.

Figure 8-6 shows the process of bottom-up service identification.

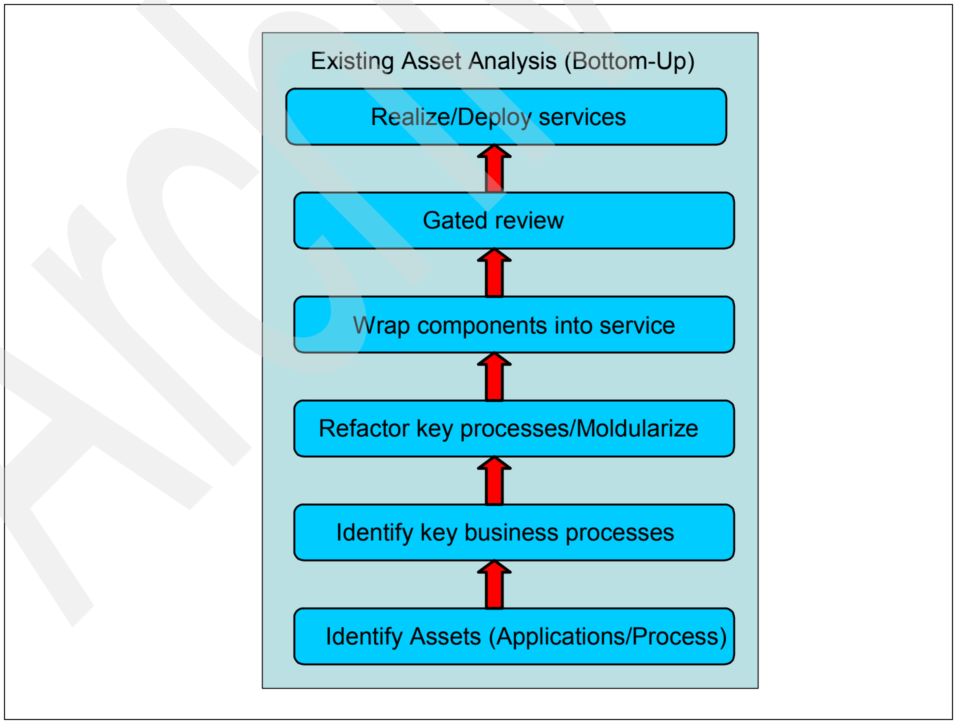

Existing Asset Analysis (Bottom-Up)
Realize/Deploy services
Gated review
Wrap components into service
Refactor key processes/Moldularize
Identify key business processes
Identify Assets (Applications/Process)

Figure 8-6 Bottom-up service identification

The likely scenario for service enablement on a z/OS platform is the refactoring of legacy applications into service components. For that reason a bottom up approach is the one most often taken. We therefore place the most emphasis on asset refactoring and the bottom up approach in this chapter.

8.4.3 Middle out service identification

The middle out approach is a combination of the top down and bottom up approaches. Both domain decomposition and existing asset analysis can take place in parallel. The results from both processes are then matched against the predetermined business goals, and services are then vetted against the governance processes to ensure usability. This tends to be a preferred method because it safeguards against the failures of top-down (project view only) as well as bottom-up (IT view only).

8.4.4 Service identification tooling

The three processes circled in Figure 8-7 on page 470 cover the following activities and matching tools:

- Discovering/understanding existing assets
  - Rational Asset Analyzer (RAA®)
    `http://www.ibm.com/software/awdtools/raa/`
  - Rational Transformation workbench (RTW)
    `http://www-01.ibm.com/software/awdtools/rtw/`
- Modeling business processes
  - WebSphere Business Modeler
    `http://www.ibm.com/software/integration/wbimodeler/`
- Modeling applications
  - Rational Software Architect (RSA)
    `http://www.ibm.com/software/awdtools/architect/swarchitect/`

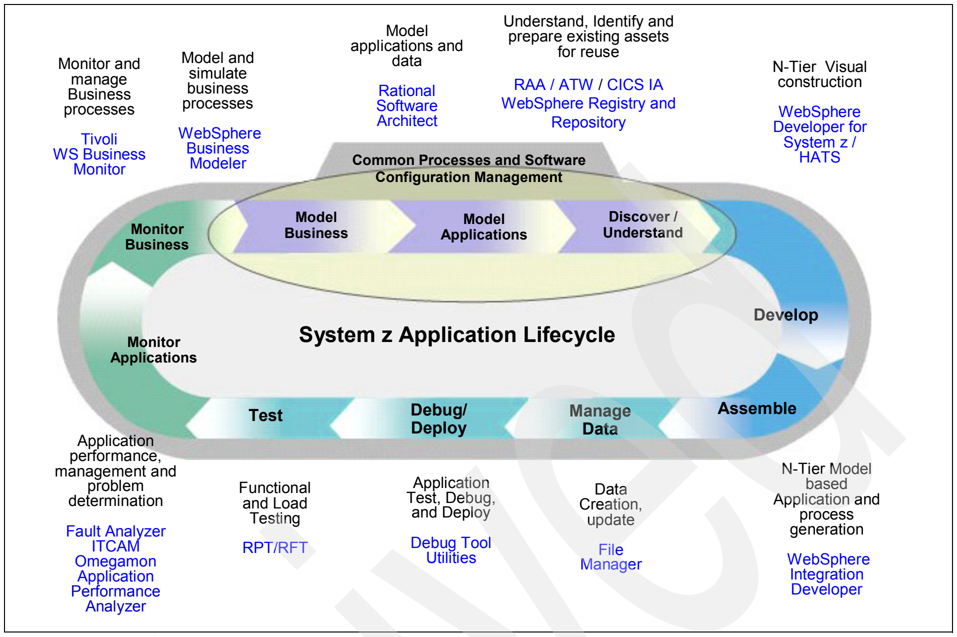

Monitor and manage Business processes
Tivoli WS Business Monitor
Model and simulate business processes
WebSphere Business Modeler
Model applications and data
Rational Software Architect
Understand, Identify and prepare existing assets for reuse
RAA / ATW / CICS IA WebSphere Registry and Repository
N-Tier Visual construction
WebSphere Developer for System z / HATS
Common Processes and Software Configuration Management
Monitor Business
Model Business
Model Applications
Discover / Understand
Develop
System z Application Lifecycle
Monitor Applications
Test
Debug/ Deploy
Manage Data
Assemble
Application performance, management and problem determination
Fault Analyzer ITCAM Omegamon Application Performance Analyzer
Functional and Load Testing
RPT/RFT
Application Test, Debug, and Deploy
Debug Tool Utilities
Data Creation, update
File Manager
N-Tier Model based Application and process generation
WebSphere Integration Developer

Figure 8-7 Tools of importance to service identification

Important: When reusing existing assets for new services it is not only important to gain insight into existing assets, it is also important to rework those existing assets back into models if they are going to be used in new services. In the next iteration of updates to the service we need to start working on the models and not the code.

Asset analysis tooling

In order to refactor an existing application it is necessary to have an in-depth understanding of the exact nature of the application. This refactoring process really has to be accomplished using tools. In the following sections we discuss the most important tools for this stage.

Rational Asset Analyzer

Rational Asset Analyzer (RAA) is a tool that provides capabilities to scan existing legacy environments for components and build an inventory. It also provides functionality for extensive impact analysis and gaining application insight.

Figure 8-8 on page 471 shows the main functionality of RAA.

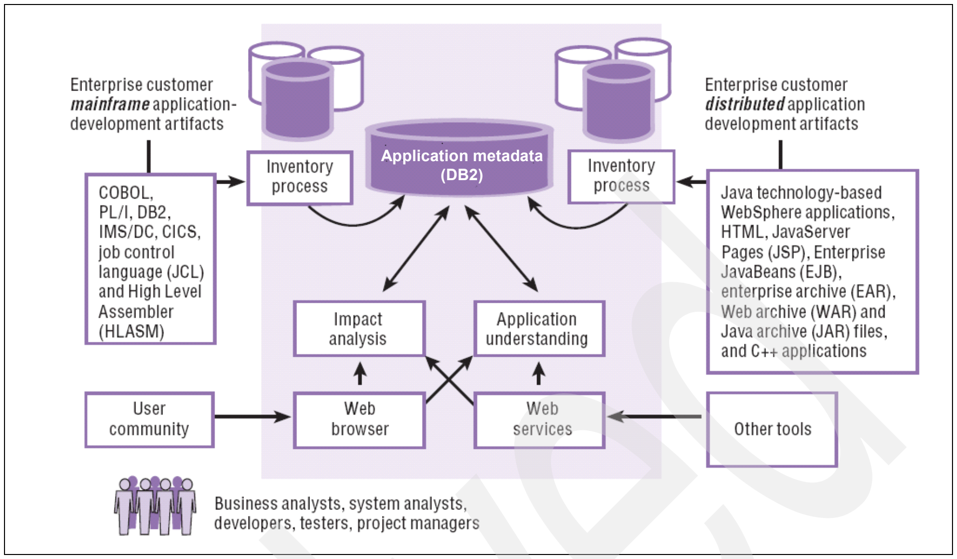

Enterprise customer *mainframe* application-development artifacts
COBOL, PL/I, DB2, IMS/DC, CICS, job control language (JCL) and High Level Assembler (HLASM)
Inventory process
Application metadata (DB2)
Inventory process
Enterprise customer *distributed* application development artifacts
Java technology-based WebSphere applications, HTML, JavaServer Pages (JSP), Enterprise JavaBeans (EJB), enterprise archive (EAR), Web archive (WAR) and Java archive (JAR) files, and C++ applications
Impact analysis
Application understanding
User community
Web browser
Web services
Other tools
Business analysts, system analysts, developers, testers, project managers

Figure 8-8 Rational Asset Analyzer (RAA) functions

Figure 8-9 shows the standard configuration of WebSphere Studio Asset Analyzer (WSAA), RAA's predecessor, on z/OS.

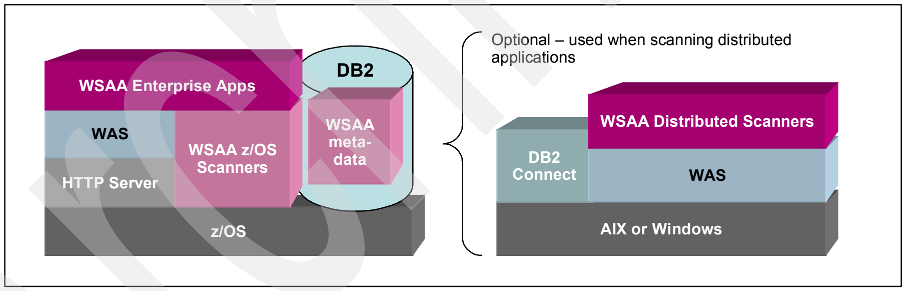


Figure 8-9 WebSphere Studio Asset Analyzer (WSAA) on z/OS

Rational Asset Analyzer is a static analysis tool that provides a developer or an architect with a high level view of the application inventory, and optionally allows them to drill down into the application to examine its structure and determine component inter-relationships. Dependencies between the application artifacts can be observed, and those dependencies can be used to determine the level of effort needed to make appropriate modifications or rewrites to the application for SOA enablement.

RAA is referred to as a "static analysis" tool because it only analyzes what the administrator tells it to — there is no awareness of running applications. If a partitioned data set filled with source code is fed into RAA, it will analyze it, even if that code has not actually executed in decades. Dynamic tools, such as the CICS Interdependency Analyzer, collect data on running systems and provide information about components actually in use. However, this information is not as complete as that provided by RAA (since source and other related artifacts are not available at run time). In addition, dynamic tools might not catch information about programs that run only on occasion, such as monthly or year-end jobs.

Note that RAA is not restricted to host assets; it can also examine and report on artifacts from Java, C++, and other Web and distributed applications. The artifacts are fed to the RAA inventory process engine, RAA adds the artifacts to the DB2-based repository, and then the user, via either a Web browser or Web services interface, can perform inquiries and analysis on those artifacts added to the repository.

The results of a typical RAA query are shown in Figure 8-10. Here, a batch job is decomposed, showing the job steps and the DD names (files) used by the job. If possible, the sub-components are hot-linked so the user can drill down and examine the various parts of the job (programs, files, databases, and so forth).

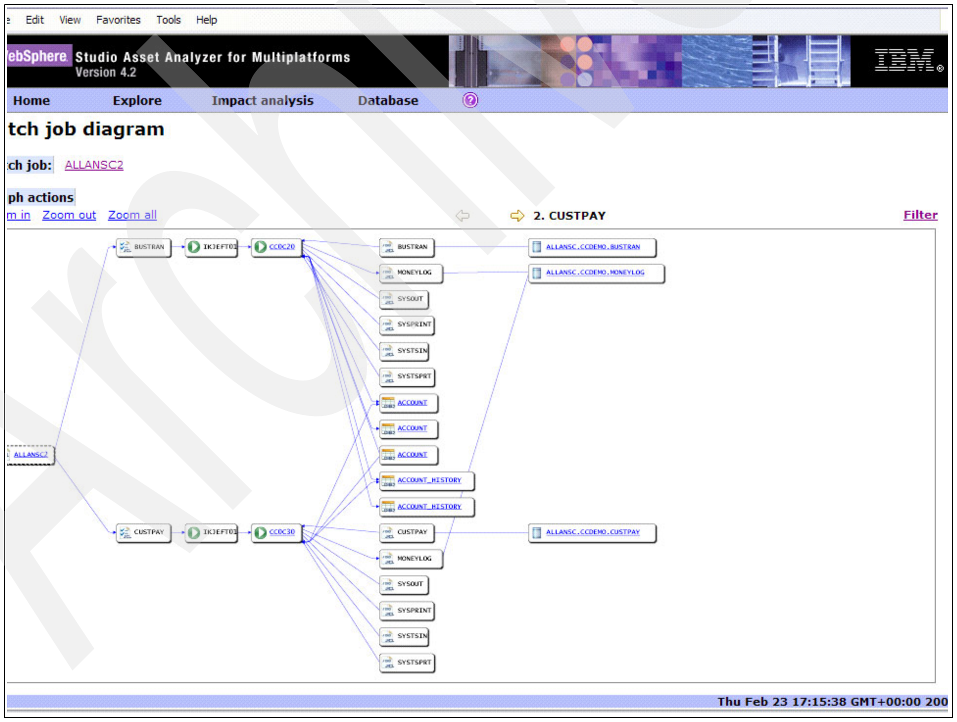


Figure 8-10   Batch job analysis in WebSphere Studio Asset Analyzer

A moderate amount of detail about program content, including the ability to browse source code, is available through RAA, which is intended to provide a high level view of the application structure and interdependencies between components. However, detailed analysis and decomposition of program assets is better accomplished in the Asset Transformation Workbench, discussed in the next section.

Rational Transformation Workbench

One tool of particular interest in refactoring is IBM Rational Transformation Workbench (RTW). This tool provides capabilities to analyze code, but also to actually refactor code. It also contains mining functionality for service identification based on existing assets. Within RTW, applications are imported into a repository called the *Enterprise Application knowledge base*. From here, hidden knowledge about business rules and processes can be quickly analyzed and refactored. The knowledge base of application information provided by RTW can help reduce risks and costs incurred with enterprise application modernization initiatives such as moving to an SOA environment.

Figure 8-11 gives an overview of the functionality of RTW.

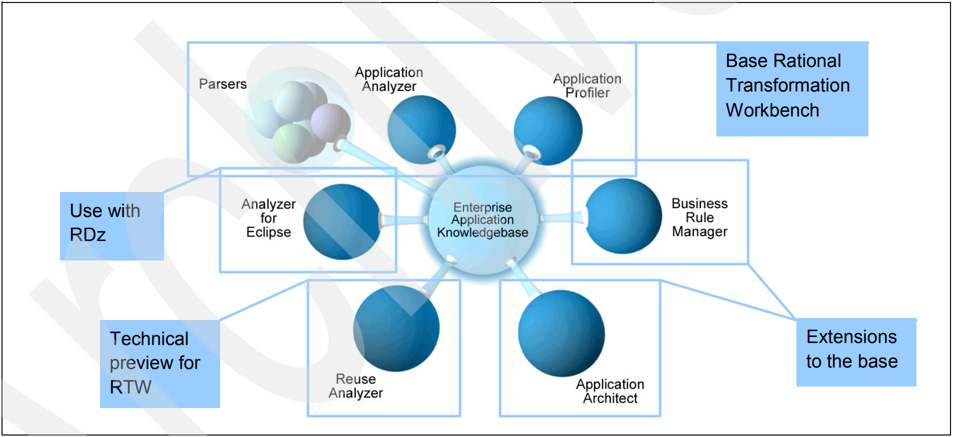
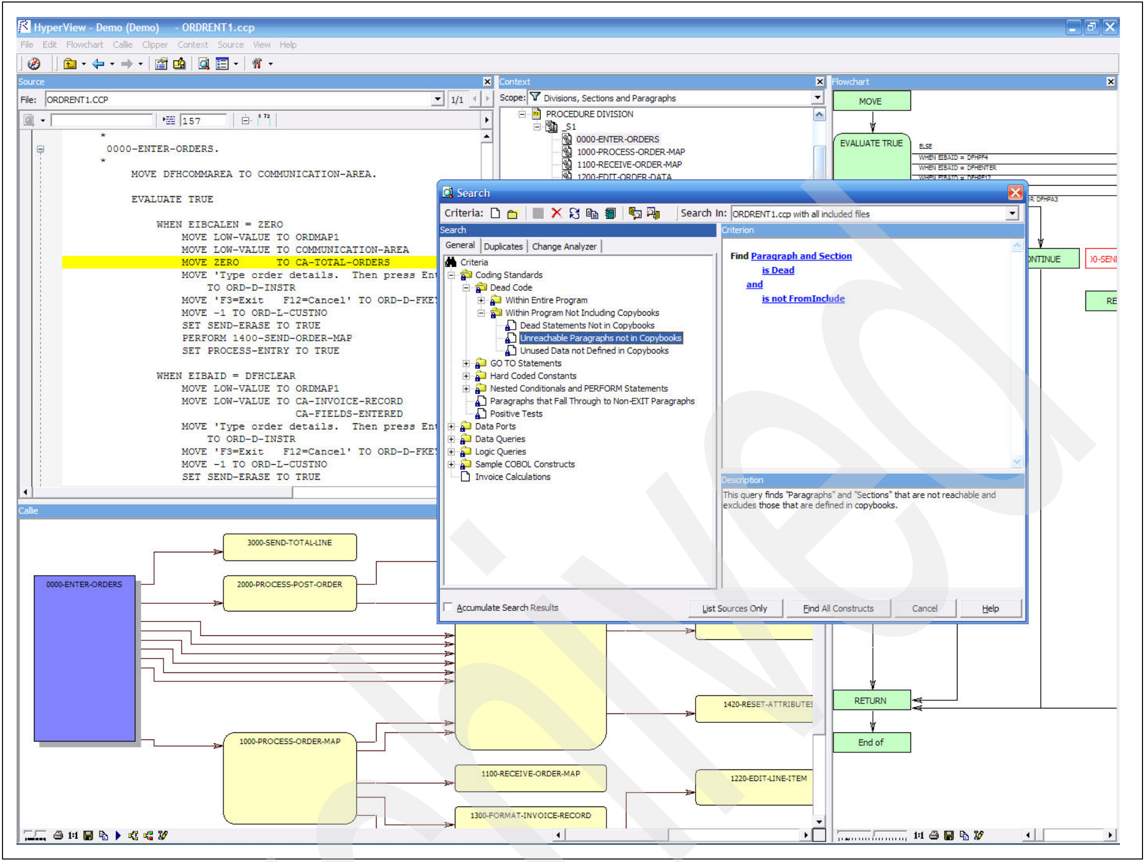


Figure 8-11 Rational Transformation Workbench (RTW) functions

Figure 8-12 on page 474 shows an example of the tool's interface.



```
      *
       0000-ENTER-ORDERS.
      *
           MOVE DFHCOMMAREA TO COMMUNICATION-AREA.

           EVALUATE TRUE

               WHEN EIBCALEN = ZERO
                   MOVE LOW-VALUE TO ORDMAP1
                   MOVE LOW-VALUE TO COMMUNICATION-AREA
                   MOVE ZERO      TO CA-TOTAL-ORDERS
                   MOVE 'Type order details.  Then press Ent
                       TO ORD-D-INSTR
                   MOVE 'F3=Exit   F12=Cancel' TO ORD-D-FKEY
                   MOVE -1 TO ORD-L-CUSTNO
                   SET SEND-ERASE TO TRUE
                   PERFORM 1400-SEND-ORDER-MAP
                   SET PROCESS-ENTRY TO TRUE

               WHEN EIBAID = DFHCLEAR
                   MOVE LOW-VALUE TO ORDMAP1
                   MOVE LOW-VALUE TO CA-INVOICE-RECORD
                                     CA-FIELDS-ENTERED
                   MOVE 'Type order details.  Then press Ent
                       TO ORD-D-INSTR
                   MOVE 'F3=Exit   F12=Cancel' TO ORD-D-FKEY
                   MOVE -1 TO ORD-L-CUSTNO
                   SET SEND-ERASE TO TRUE
```

Find Paragraph and Section
is Dead
and
is not FromInclude

This query finds "Paragraphs" and "Sections" that are not reachable and excludes those that are defined in copybooks.

Figure 8-12 Rational Transformation Workbench

Rational Transformation Workbench (RTW) provides detailed reports, metrics, and visualizations of existing mainframe applications. The foundation of RTW is a knowledge base that contains information that describes the applications. Surrounding the knowledge base are a number of key capabilities and features[1]:

- Detailed reports, metrics, documentation, and visualizations of the enterprise applications are readily accessible to project leaders and architects using the workbench.
- A browser-based module to allow team members to use RTW-generated reports.
- Integration with IBM Rational Studio Asset Analyzer to allow users to perform high level analysis in WebSphere Studio Asset Analyzer and pass the

[1] Complete feature descriptions for ATW can be found at
http://www-306.ibm.com/software/awdtools/rtw/features/

application insight through a software bridge for use in Asset Transformation Workbench.

- Powerful analysis and assessment tools to help accelerate ongoing maintenance and enhancements.
- Tools to expose and help manage business rules, which can simplify application reuse for SOA initiatives.
- Re-architecting tools to help increase the productivity of teams restructuring and componentizing applications.
- A Reuse Analyzer for RTW (technical preview) to help quickly assess an application's suitability for reuse in an SOA.

There are several main components of RTW that use the knowledge base:

- **Application Analyzer**
  The Application Analyzer is a non-invasive interactive module that creates a comprehensive repository of system relationships including source code, system files, data definition language (DDL), screen maps, and more. It can help perform impact analysis, generate interactive graphical system diagrams, create system documentation, and browse source code in context-sensitive mode.
- **Application Profiler**
  The Application Profiler provides technical and business users with information to effectively understand and plan enterprise applications, without impacting the source code. It is a browser-based tool that provides users such as support, quality assurance, and business analysts insight into systems without requiring specialized knowledge or skills. Technical users who are unfamiliar with their enterprise applications can use Application Profiler to access documentation, understand system structure, and determine the impact of code changes. Non-technical or business users can use Application Profiler to examine system-level reports and to assess where to direct resources in order to enhance or renovate the application portfolio. When used with the Business Rules Extension, the Application Profiler module can help organize and annotate business rules without disturbing development.
- **Business Rules Extension**
  The Business Rules feature is an *optional* extension that helps navigate complex code and identify, document, and organize business rules. It identifies candidate rules using developer-driven sophisticated search algorithms. This process generates a list of rules for the targeted application, allowing analysts to view each rule and verify its inclusion. After rules have been found they can be documented and organized, allowing future users to understand the use of each rule. And because the rules are tagged, analysts

can locate the rules within the code and modify them to respond to business process changes.

- **Application Architect**
  The Application Architect feature is an *optional* extension that uses sophisticated algorithms to partition code into new components. The componentization of logic results in a structured architecture that can reduce complexity and facilitate modernization. By componentizing enterprise code, developers are able to greatly increase the performance of frequently used programs. Application Architect can help to ensure that the components created are complete, working programs in accordance with the functionality of the original application.
- **Reuse Analyzer**
  The Reuse Analyzer extension (in RTW Version 3.1, a "technology preview") can:
  - Categorize CICS and IMS programs written in COBOL by the type of work they do (screen, business logic, data access, hybrid, and so forth).
  - Identify some potential architectural "traps" that would require remediation before making a particular program or program call hierarchy available as a Web service.
  - Create Web Services Description Language (WSDL) files corresponding to selected data elements in your program you wish to make available in a Web service. (A WSDL file can then be used with XML Services for the Enterprise, which is a feature of WebSphere Developer for zSeries.)

Rational Transformation Workbench is used when a detailed view of the application is needed for work on re-engineering an application. Rational Asset Analyzer is used when a higher level view of the application assets is needed.

RTW is a participant in the Model stage of the SOA life cycle, but it also has some participation in the Assemble stage, where it provides key features, such as the Application Architect and Reuse Analyzer, that assist in the actual code creation.

For more information about the capabilities of RTW see the IBM publications and Web site:

- *IBM Rational Transformation Workbench Getting Started,* SC31-6877
- *IBM Rational Transformation Workbench Analyzing Programs* - SC31-6877
- `http://www.ibm.com/software/awdtools/rtw/`

Tools used for modeling

From an application rejuvenation perspective, the artifacts that are produced from the Model stage of the SOA life cycle are primarily process models and service definitions that come from the modeling tools such as WebSphere Business Modeler, WebSphere Integration Developer (WID), and Rational Software Architect (RSA).

Tooling allows you to model processes, decision points, choreography, and relationships within an application. Although most tooling is often provided for Object Oriented based applications, the basis for this modeling is usually *Unified Modeling Language (UML)*.

While UML is still used for these purposes, other more coarse grained, business focused modeling tools are now in use for business modeling in SOA. WebSphere Business Modeler is IBM's key product for creating models of business processes and coarse grained composite transactions.

Rational RequisitePro

In addition to modeling processes and applications, the developer and architect must also manage requirements. IBM's Rational RequisitePro® tool can be used for requirements gathering and management, to feed the process of re-engineering existing code assets or for construction of new services.

8.5 Service categorization

This is the next logical step after service identification. Service categorization requires a service to be associated within a hierarchical domain. These domains might be business, infrastructure, and support. They might also be associated with the business domain they relate to, such as customer, loans, and so forth. The first benefit from this process is ease of discovery. Potential consumers can browse a service catalogue according to the domain they relate to. It also reduces any ambiguity should service naming standards not have been adhered to at some stage. A second benefit is the prevention of service duplication. Categorizing services into domains should aide in providing a clearer view of what is and is not available within a domain.

Ideally, this information is provided in a single place. This makes it is easy to search asset repositories for reusable services and to get all of the necessary information without having to navigate many different documents or search for related elements. Service specifications include at least this information:

- The name of the service, suggesting its purpose.
- The provided and required interfaces, thereby defining the functional capabilities that are provided by the service and those that it requires of its consumers. Note that this is not about how the service is implemented, but rather the interaction between the consumers and providers for this service.

- Any protocol that specifies rules for how the functional capabilities are used or in what order.
- Constraints that reflect what successful use of the service is intended to accomplish and how it will be evaluated.
- Qualities that service consumers should expect and that providers are expected to provide, such as cost, availability, performance, footprint, suitability to the task, competitive information, and so forth.
- Policies for using the service, such as security and transaction scopes for maintaining security and integrity or for recovering from the inability to successfully perform the service or any required service.

8.6 Tools for service life cycle management

After services have been identified, either as a result of a top down or bottom up approach, the development process can begin. This process includes a number of phases, shown in Figure 8-13.

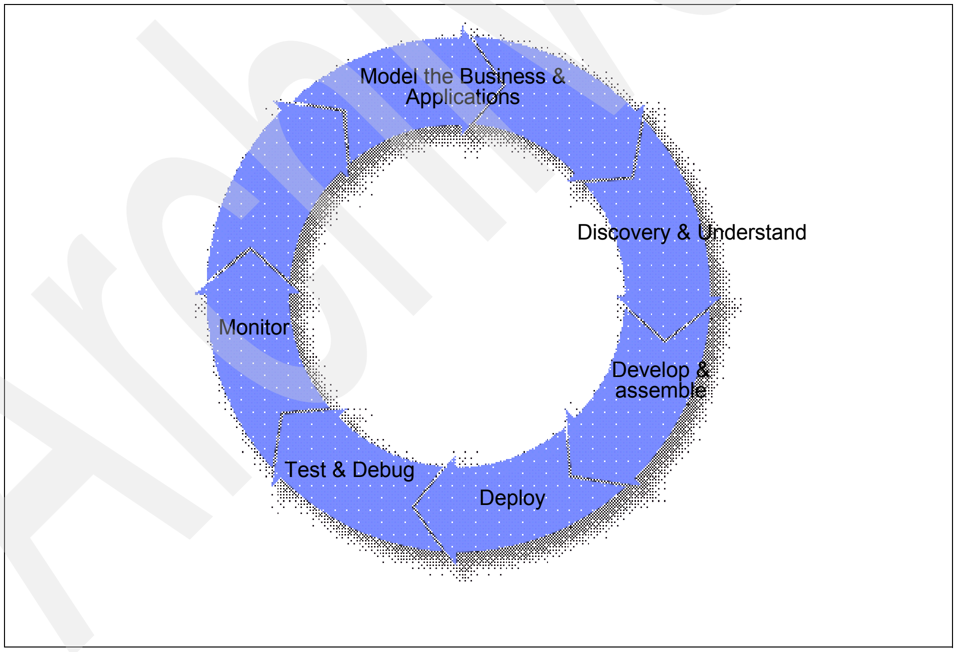
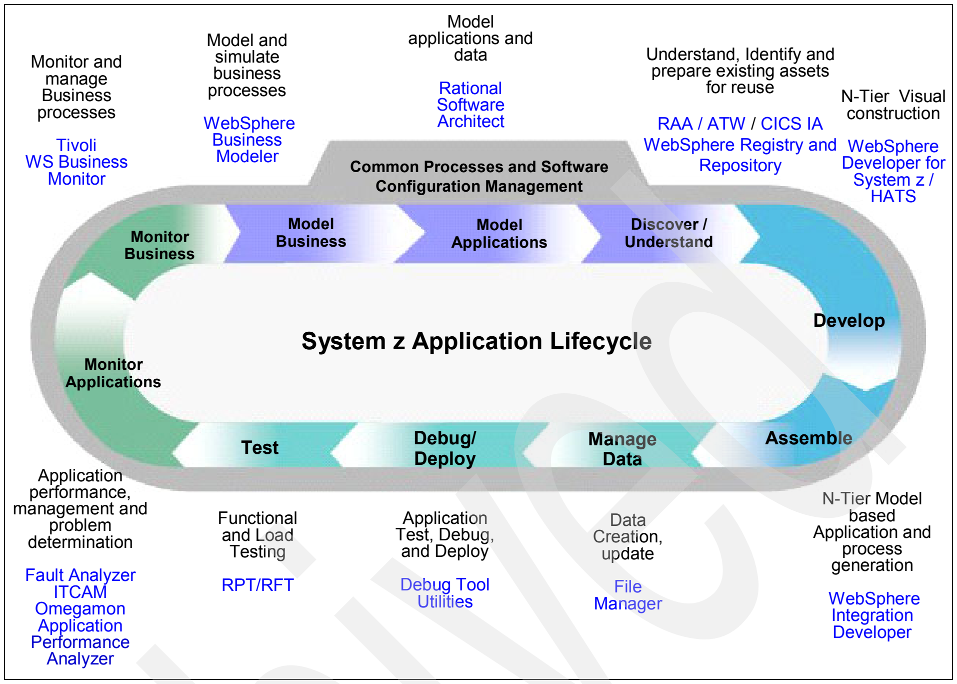


Figure 8-13 Service life cycle

The IBM tools offered for each phase are shown in Figure 8-14 on page 479.



Monitor and manage Business processes
Tivoli WS Business Monitor
Model and simulate business processes
WebSphere Business Modeler
Model applications and data
Rational Software Architect
Understand, Identify and prepare existing assets for reuse
RAA / ATW / CICS IA
WebSphere Registry and Repository
N-Tier Visual construction
WebSphere Developer for System z / HATS
Common Processes and Software Configuration Management
Monitor Business
Model Business
Model Applications
Discover / Understand
Develop
System z Application Lifecycle
Monitor Applications
Test
Debug/ Deploy
Manage Data
Assemble
Application performance, management and problem determination
Fault Analyzer ITCAM Omegamon Application Performance Analyzer
Functional and Load Testing
RPT/RFT
Application Test, Debug, and Deploy
Debug Tool Utilities
Data Creation, update
File Manager
N-Tier Model based Application and process generation
WebSphere Integration Developer

Figure 8-14 Service life cycle tools

Refer to the publication *Building SOA Solutions Using the Rational SDP*, SG24-7356 for a good description of most of the Rational tools identified in the diagram.

A

Business Process services application patterns

Business Process patterns are observed where multiple automated business processes are combined to yield a new business offering or to provide a consolidated view of some business entity with many representations in the corporate business systems. Process-focused patterns are used to combine multiple business processes or business systems. The result is a new business offering or a consolidated view of some business entity. These patterns are particularly useful in tying together different platforms and technologies. However, they represent a more difficult design and development task compared to data integration and often require complex middleware.

Business and IT drivers that result in the selection of an Integration pattern are:

- The business processes must be integrated with existing business systems and information.
- The business activity must aggregate, organize, and present information from various sources within the organization.

The pattern does not include front-end integration such as the composition of a portal or single sign-on across multiple applications; they are captured by the other patterns.

The following patterns are discussed:

- Serial Process pattern
- Exposed Serial Process pattern
- Parallel Process pattern
- Serial Workflow variation
- Exposed Serial Workflow pattern
- Parallel Workflow variation

Serial Process pattern

The Serial Process pattern, shown in Figure A-1, supports the sequential execution of business services hosted by a number of target applications. The source application initiates a serial business process.

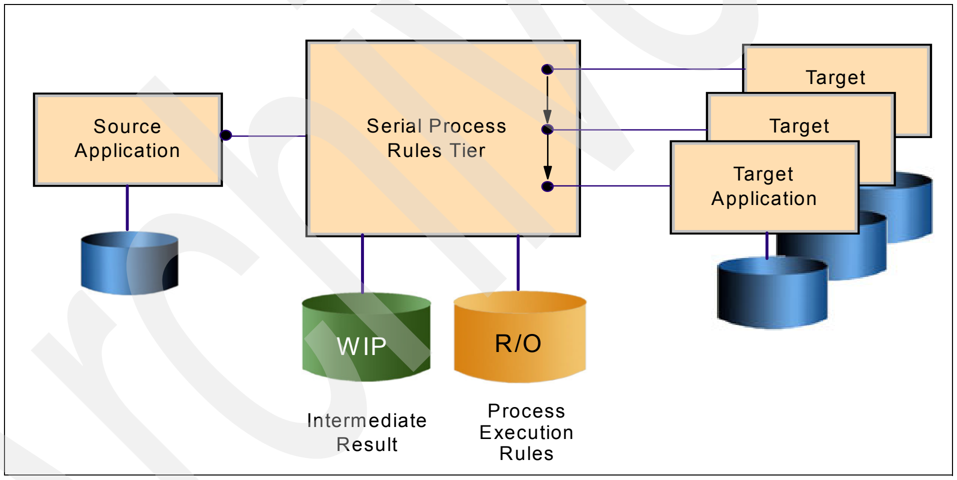


Figure A-1 Serial Process pattern

Solution

The Serial Process pattern is broken down into three logical tiers:

- The Source Application tier represents one or more applications that are interested in interacting with the target applications.
- The Serial Process Rules tier supports most of the services provided by the process tier in the pattern, including routing of requests, protocol conversion, message broadcasting, and message decomposition and recomposition. It also supports the separation of business process flow logic from individual

application logic. The process logic is governed by the serial process rules that define execution rules for each target application, together with control flow and data flow rules. It might also include any necessary adapter rules. The combination of these process execution rules are stored in read-only databases. This externalization of process flow logic is essential for the implementation of a flexible and responsive IT environment that can respond quickly to changing business needs. It also makes it possible to compose new end-to-end processes by combining different business services provided by different applications. Finally, this tier uses a work-in-progress (WIP) database to store the intermediate results from the execution of different process steps.

- The Target Application tier represents new, modified existing, or unmodified existing applications that implement the necessary business services.

Benefits

The Serial Process pattern improves the flexibility and responsiveness of an organization by implementing end-to-end process flows and by externalizing process logic from individual applications. In addition, it provides a foundation for automated support for Business Process Management that enables the monitoring and measurement of the effectiveness of business processes.

Run time

Figure A-2 shows the Serial Process runtime pattern.

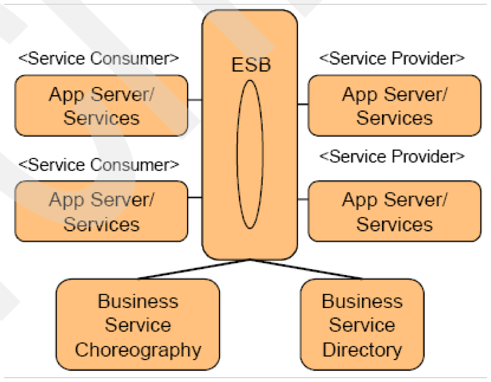


Figure A-2 Serial Process runtime pattern

Exposed Serial Process pattern

The Exposed Serial Process pattern, shown in Figure A-3, extends the one-to-N topology provided by the pattern with sequential execution of business services hosted by a number of target applications. The source application initiates a serial business process across enterprise boundaries. The Exposed Serial Process pattern separates the process logic from application logic that is distributed across organization boundaries. The process logic is governed by Serial Process Rules that define execution rules for each target application, together with control flow and data flow rules. It might also include any necessary adapter rules.

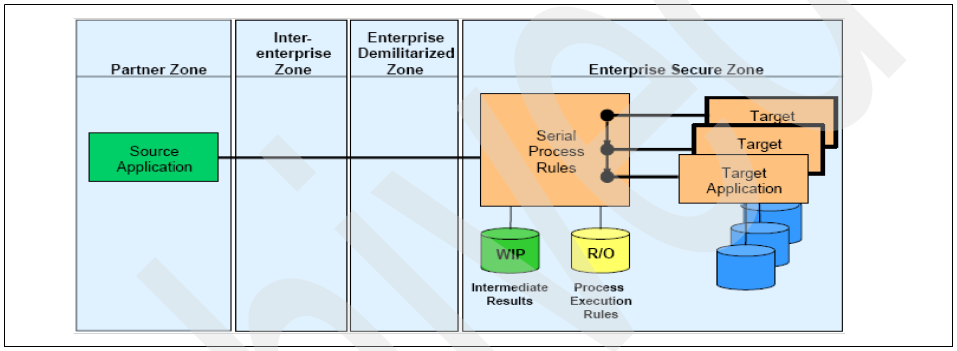


Figure A-3 Exposed Serial Process pattern

Solution

The Exposed Serial Process pattern is broken down into three logical tiers:

- The Source Application tier represents an application in another organization that is interested in interacting with the Exposed Serial Process.
- The Serial Process Rules tier supports most of the services provided by the process tier in the pattern, including the routing of requests, protocol conversion, message broadcasting, and message decomposition and recomposition. In addition, it supports the separation of business process flow logic from individual application logic. The process logic is governed by Serial Process Rules that define execution rules for each target application, together with control flow and data flow rules. It might also include any necessary adapter rules. The combination of these process execution rules is stored in read-only databases. This externalization of process flow logic is essential for the implementation of a flexible and responsive IT environment that can respond quickly to changing business needs. It also makes it possible to compose new end-to-end processes by combining different business services provided by different applications. Finally, this tier uses a WIP database to store the intermediate results from the execution of different process steps.

- The Target Application tier represents new, modified existing, or unmodified existing applications that implement the necessary business services.

Benefits

The Exposed Serial Process pattern improves the flexibility and responsiveness of an organization. It does this by implementing end-to-end process flows across organization boundaries and by externalizing process logic from individual applications. In addition, it provides a foundation for automated support for business process management that enables the monitoring and measurement of the effectiveness of business processes.

Run time

Figure A-4 shows the Exposed Serial Process runtime pattern.

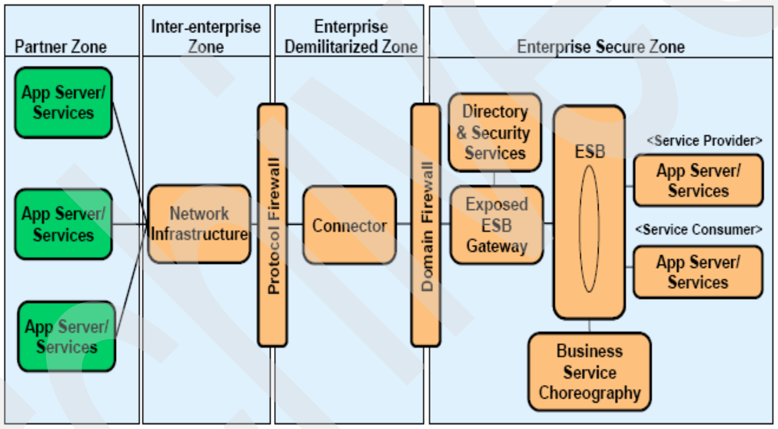


Figure A-4 Exposed Serial Process runtime pattern

Parallel Process pattern

The Parallel Process pattern, shown in Figure A-5, adds concurrency to the Serial Process pattern. The process orchestration might include paths that lead to parallel invocation of target application services.

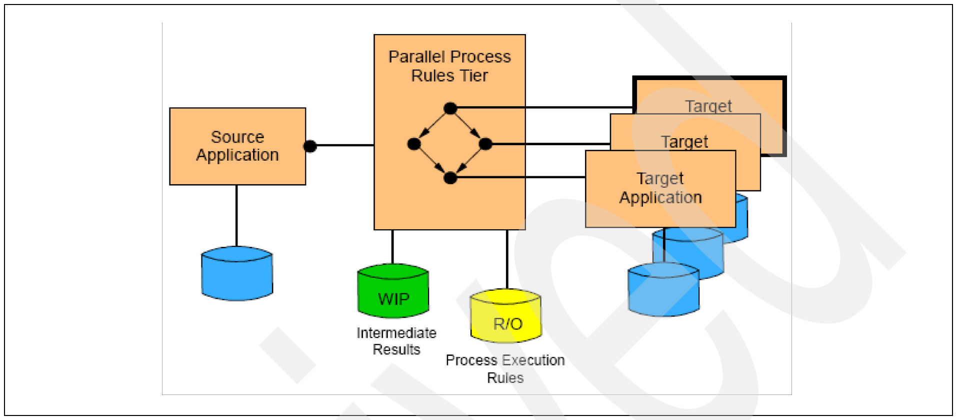


Figure A-5 Parallel Process pattern

Solution

The Parallel Process pattern is broken down into three logical tiers:

- The Source Application tier is the same as for the Serial Process pattern.
- The Parallel Process Rules tier supports all the services provided by the Serial Process Rules tier within the Serial Process pattern. In addition, the interaction initiated by the source application might invoke services on multiple target applications in parallel. This parallelism requires that fork and join conditions be defined. The runtime engine must be able to initiate parallel threads of control, ensure these threads join upon completion, and manage them as a unit (for example, to allow cancellation of the process or to report its status).
- The Target Application tier is the same as for the Serial Process pattern.

Benefits

In addition to providing all the benefits provided by the Serial Process pattern, this pattern provides a foundation for the reduction of cycle times by implementing parallel processes.

Run time

Figure A-6 on page 487 shows the Parallel Process runtime pattern.

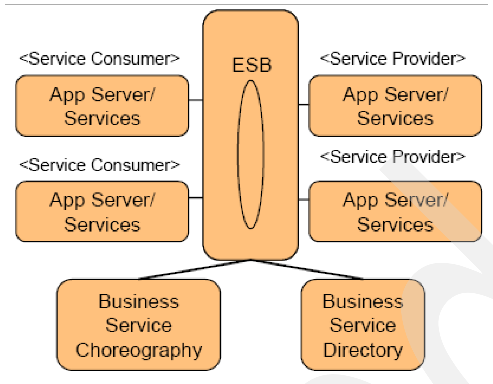
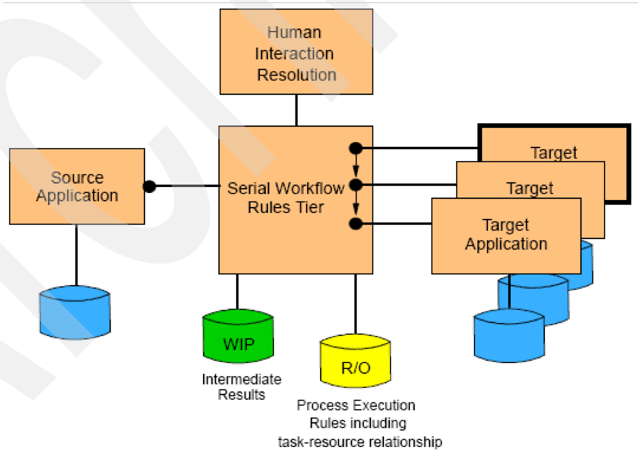


Figure A-6 Parallel Process runtime pattern

Serial Workflow variation

The Serial Workflow variation of the Serial Process pattern, shown in Figure 8-15, extends the basic Serial Process by including interaction with people as steps in the business process.



Human Interaction Resolution
Source Application
Serial Workflow Rules Tier
Target
Target
Target Application
WIP
Intermediate Results
R/O
Process Execution Rules including task-resource relationship

Figure 8-15 Serial Workflow variation

Solution

The Serial Workflow variation is broken down into three logical tiers:

- The Source Application tier is the same as for the Serial Process pattern.
- In addition to all the services provided by the Serial Process Rules tier within the Serial Process pattern, the Serial Workflow Rules tier supports routing certain tasks to human actors for completion. The rules are augmented with relationships that define the resources that are capable of performing specific tasks. People, departments, and target applications can all be resources capable of executing a particular task. This tier resolves task-resource relationships during the execution of a process. If human interaction is needed, the task is added to a work list to be completed by a human. The process is typically suspended until the completion of the task. This tier provides support for long-running transactions, using a WIP database to store intermediate results while the process runs.
- The Target Application tier is the same as for the Serial Process pattern.

Benefits

In addition to the benefits of the Serial Process pattern, further flexibility is introduced by the externalization of task-resource resolution rules, which allow people to execute steps in the process.

Run time

Figure A-7 on page 488 shows the Serial Workflow variation run time.

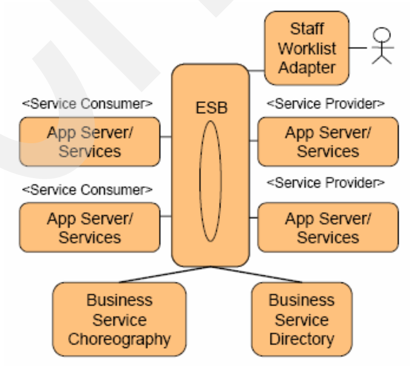


Figure A-7 Serial Workflow variation run time

Exposed Serial Workflow pattern

The Workflow variation of the Exposed Serial Process pattern, shown in Figure A-8, extends the basic Serial Process orchestration capability by supporting human interaction for completing certain process steps.

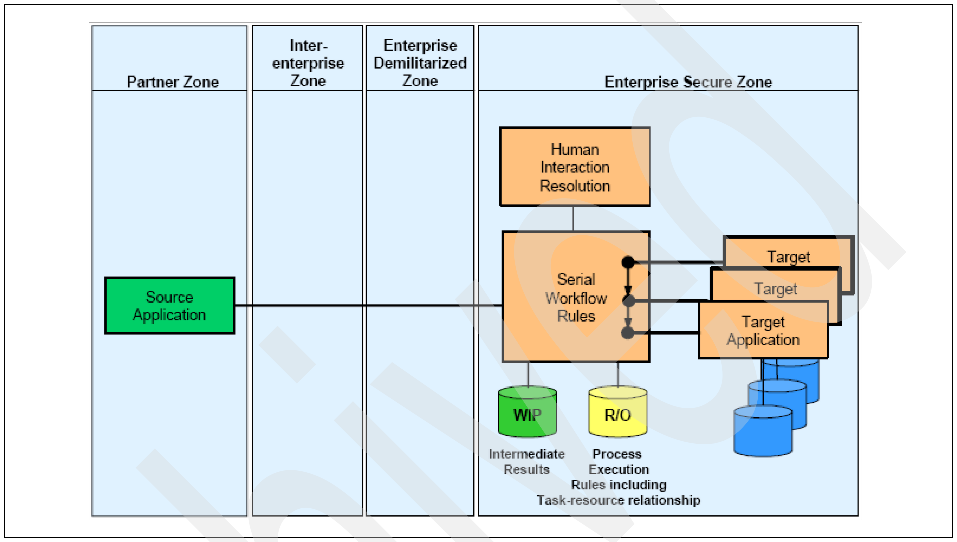


Figure A-8 Exposed Serial Workflow pattern

Solution

The Serial Workflow variation is broken down into three logical tiers:

- The Source Application tier is the same as for the Exposed Serial Process pattern.
- The Serial Workflow Rules tier supports all the services provided by the Serial Process Rules tier within the Exposed Serial Process pattern. In addition, it supports certain tasks within the process to be routed to a person or people for completion. The rules are augmented with relationships that define the resources that are capable of performing specific tasks. People, departments, and target applications can all be resources capable of executing a particular task. This tier resolves task-resource relationships during the execution of a process. If human interaction is needed, the task is added to a worklist to be completed by a human. The process is typically suspended until the completion of the task. This tier provides support for long-running transactions, using a WIP database to store intermediate results while the process runs.

- The Target Application tier is the same as for the Exposed Serial Process pattern.

Benefits

In addition to the benefits of the Exposed Serial Process pattern, further flexibility is introduced by the externalization of task-resource resolution rules, which allow people to perform steps in the process.

Run time

Figure A-9 shows the Exposed Serial Workflow runtime pattern.

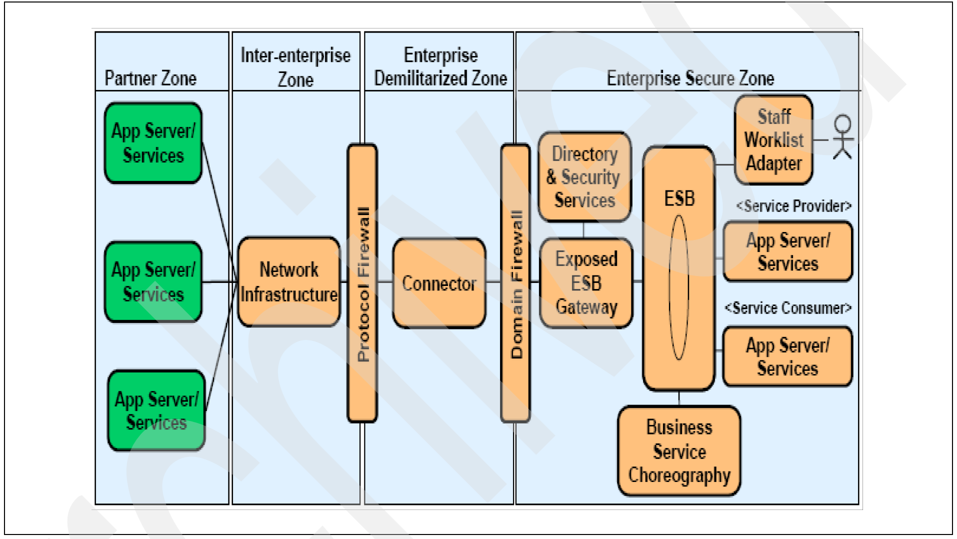


Figure A-9 Exposed Serial Workflow runtime pattern

Parallel Workflow variation

The Parallel Workflow variation of the Parallel Process pattern, shown in Figure A-10 on page 491, extends the basic parallel process orchestration capability by supporting human interaction for completing certain process steps.

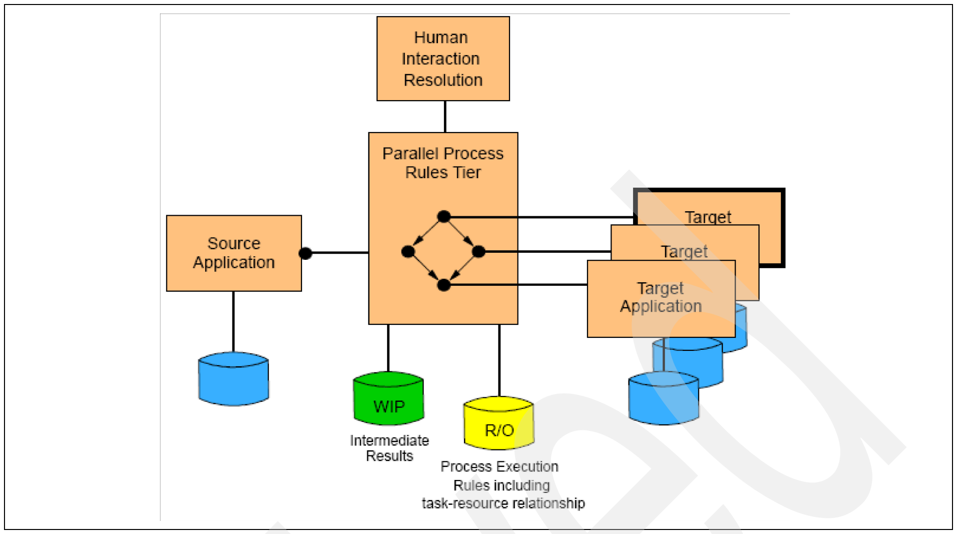

Human Interaction Resolution
Parallel Process Rules Tier
Source Application
Target
Target
Target Application
WIP
Intermediate Results
R/O
Process Execution Rules including task-resource relationship

Figure A-10 Parallel Workflow variation

Solution

The Parallel Workflow variation is broken down into three logical tiers:

- The Source Application tier is the same as for the Parallel Process pattern.
- The Parallel Workflow Rules tier supports all the services provided by the Parallel Process Rules tier within the Parallel Process pattern. In addition, it supports routing certain tasks to people for completion. The rules are augmented with relationships that define the resources that are capable of performing specific tasks. People, departments, and target applications can all be resources capable of executing a particular task. This tier resolves task-resource relationships during the execution of a process. If human interaction is needed, the task is added to a worklist to be completed by a human. The process is typically suspended until the completion of the task. This tier provides support for long-running transactions, using a WIP database to store intermediate results while the process runs.
- The Target Application tier is the same as for the Parallel Process pattern.

Benefits

In addition to the benefits of the Parallel Process pattern, further flexibility is introduced by the externalization of task-resource resolution rules, which allow people to execute steps in the process.

Run time

Figure A-11 on page 492 shows the Parallel Workflow variation run time.

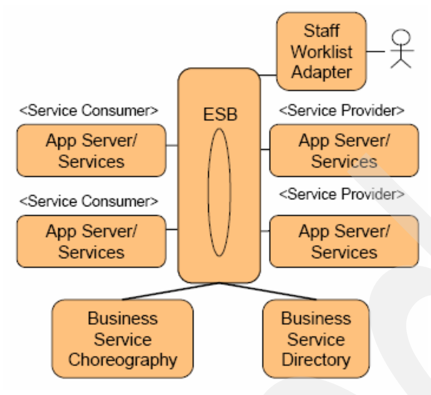


Figure A-11 Parallel Workflow variation runtime

Summary

The following Process-focused patterns were discussed:

- Serial Process pattern
- Exposed Serial Process pattern
- Parallel Process pattern
- Serial Workflow variation
- Exposed Serial Workflow pattern
- Parallel Workflow variation

The Serial Process pattern, described in "Serial Process pattern" on page 482, adds sequencing to the one-to-N topology of the pattern. It enables the orchestration of a serial business process. Its Serial Workflow variation, defined in "Serial Workflow variation" on page 487, adds support for including interaction with people as steps of the serial business process.

The Parallel Process pattern, described in "Parallel Process pattern" on page 486, adds concurrency to the Serial Process application pattern. The process orchestration might include paths that lead to parallel invocation of target application services. Its Parallel Workflow variation, specified in "Parallel Workflow variation" on page 490, adds support for including interaction with people as steps of the parallel business process.

The Exposed Serial Process pattern, described in “Exposed Serial Process pattern” on page 484, adds sequencing to the one-to-N topology of the pattern. It enables the orchestration of a serial business process across enterprise boundaries. Its Exposed Serial Workflow variation, specified in “Exposed Serial Workflow pattern” on page 489, adds support for including interaction with people as steps of the serial business process.

The Exposed Parallel Process and Exposed Parallel Workflow pattern is another possibility, but it is not currently being observed in the Extended Enterprise domain. We expect it to appear as the technology evolves.

For more information, see the Patterns for e-business Web site at:

```
http://www.ibm.com/developerWorks/patterns/
```

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 publications

For information about ordering these publications, see “How to get Redbooks publications” on page 496. Note that some of the documents referenced here might be available in softcopy only.

- *Patterns: Implementing an SOA Using an Enterprise Service Bus*, SG24-6346
- *WebSphere for z/OS V6 Connectivity Handbook* , SG24-7064
- *WebSphere Message Broker Basics*, SG24-7137
- *Implementing an ESB using IBM WebSphere Message Broker V6 and WebSphere ESB V6 on z/OS*, SG24-7335
- *SOA Transition Scenarios for the IBM z/OS Platform*, SG24-7331
- *Implementing IBM Content Manager OnDemand Solutions with Case Studies*, SG247511
- *Understanding SOA Security Design and Implementation*, SG24-7310
- *Security in WebSphere Application Server V6.1 and J2EE 1.4 on z/OS*, SG24-7384
- *SOA Foundation Service Creation Scenario*, SG24-7240
- *Implementing CICS Web Services*, SG24-7206
- *Building SOA Solutions Using the Rational SDP*, SG24-7356
- *IBM System z Strengths and Values*, SG24-7333

Other publications

These publications are also relevant as further information sources:

- *MS Message Format Services Reversal Utilities for z/OS User's Guide*, SC27-0823
- *IBM Asset Transformation Workbench Getting Started,* SC31-6877
- *IBM Asset Transformation Workbench Analyzing Programs,* SC31-6877

Online resources

These Web sites are also relevant as further information sources:

- WebSphere Process Server components

  `http://www.ibm.com/software/integration/wps/library/`
- Patterns for e-business

  `http://www.ibm.com/developerWorks/patterns/`
- WebSphere Business Process Management

  `http://www-01.ibm.com/software/websphere/products/businessint/`
- WS-AtomicTransaction specification

  `http://www.ibm.com/developerworks/library/ws-atomtran/`
- WS-BusinessActivity specification

  `http://www.ibm.com/developerworks/webservices/library/ws-busact/`
- Transactions in the world of Web Services, part 1 and part 2

  `http://www.ibm.com/developerworks/webservices/library/ws-wstx1/`
  `http://www.ibm.com/developerworks/webservices/library/ws-wstx2/`

How to get Redbooks publications

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

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

Exploiting IBM System z in a Service-Oriented Architecture

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Exploiting IBM System z in a Service-Oriented Architecture

Modernizing application environments

Using IBM SOA Reference Architecture software

Leveraging System z quality of service

There are many options to implement a service-oriented architecture (SOA). A good SOA solution for one company might not be a good solution for another, even in the same business environment and IT landscape. Choosing a solid SOA solution involves strategy, vision, architectural thinking, and finally, technology. It also involves personal taste and organizational politics.

So, why did we decide to write a book dedicated to SOA solutions on System z then? The answer is quite simple: the combination of the System z hardware, the operating systems that run on System z, and the SOA middleware provide specific functionality and influence the effectiveness of your SOA solution to a great extent. In other words, if you were to implement the same SOA solution on System z and on another platform, you would see different results.

This book projects a large number of solutions in all areas of the IBM SOA Reference Architecture on IBM System z. Where appropriate we highlight the additional System z benefits of running a certain solution on System z.

The primary audience for this book is IT Architects, especially solution architects, enterprise architects, infrastructure architects, and application architects.

SG24-7651-00 ISBN 0738432008