



WebSphere software

Deploying CICS Web services to preserve IT investments in the banking industry.

*By Nigel Williams, Certified IT Specialist, IBM Design Center,
Montpellier, France and Steve Wall, IT specialist, System z
Benchmark Center, Montpellier, France*

Contents

2 Introduction

3 Two models for CICS SOA integration

4 Customer goals

8 Usage scenarios

12 Building a high-availability SOA infrastructure

13 CICSplex capabilities

14 Parallel Sysplex capabilities

15 Validating the solution

26 Conclusion

27 For more information

Introduction

The IBM SOA reference architecture encourages the reuse of existing IT investment and promotes the reuse of services in current and future systems. Traditional mainframe IBM CICS® applications are among the most valuable assets of many large companies, particularly in the financial sector. Without these COBOL and PL/I programs, many large companies could not do business. Reusing these proven assets can result in delivering new services much more quickly, and it is significantly less expensive to reuse existing applications than to write new ones.

CICS applications can be exposed as services with well-defined interfaces. Such services enable loose coupling, providing the necessary flexibility to build composite applications, thereby improving interoperability and reuse.

In this paper we describe a practical example of how you can integrate CICS applications within a service oriented architecture (SOA) infrastructure using the Web services support provided by IBM CICS Transaction Server (CICS TS), Version 3. The paper is based on a project conducted by the IBM Customer Center (PSSC) in Montpellier, France working with a large financial services customer. We document the solution that we designed, and explain how it meets the specific requirements of this customer. We discuss some of the design decisions that were made based on the customer's requirements, and provide a description of the infrastructure that we created to test the solution. The tested infrastructure was based on CICS Transaction Server, IBM WebSphere® Application Server, IBM Parallel Sysplex® technology, the IBM WebSphere DataPower® Integration Appliance XI50 and IBM Tivoli® monitoring software.

Two models for CICS SOA integration

Figure 1 shows the two basic models that can be used to enable a CICS application as a Web service: the *indirect* and *direct* exposure models.

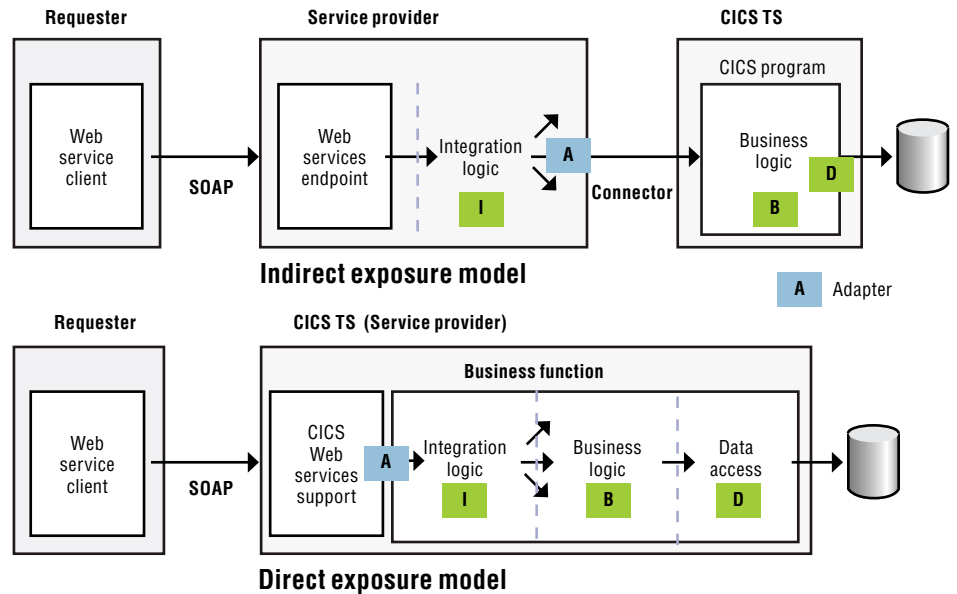


Figure 1. Two models of service-enabling CICS

In the indirect exposure model, the service endpoint is external to CICS. The service provider in this case (for example, a Java™ 2 Platform, Enterprise Edition [J2EE] application running in WebSphere Application Server) hosts the integration logic and uses an adapter to connect to the CICS system. The most common adapter of this type today is IBM CICS Transaction Gateway.

In the direct exposure model, the service endpoint is inside CICS. The service provider in this case is a procedural language program. CICS adapts the SOAP message format to the message format used by the existing CICS programs for integration and business logic (typically, a CICS communication area [COMMAREA] or container format). The direct model is enabled using the CICS Web services support provided by CICS Transaction Server, V3.1 or V3.2.

In this project we used the direct exposure model for reasons that will be explained in the following section.

Customer goals

The certified architects and specialists at the Design Center² in Montpellier help customers with leading-edge SOA projects, by running design workshops and proofs of concept. This document describes a recently completed customer project run by the Design Center and the IBM System z™ Benchmark Center at the IBM Customer Center in Montpellier, France.

Customer background

The customer is a large European financial services group that incorporates several different, well-known brands. Figure 2 shows the customer organization that handles business service requests from a number of different requesters, including branch offices, customers, suppliers and trusted business partners.

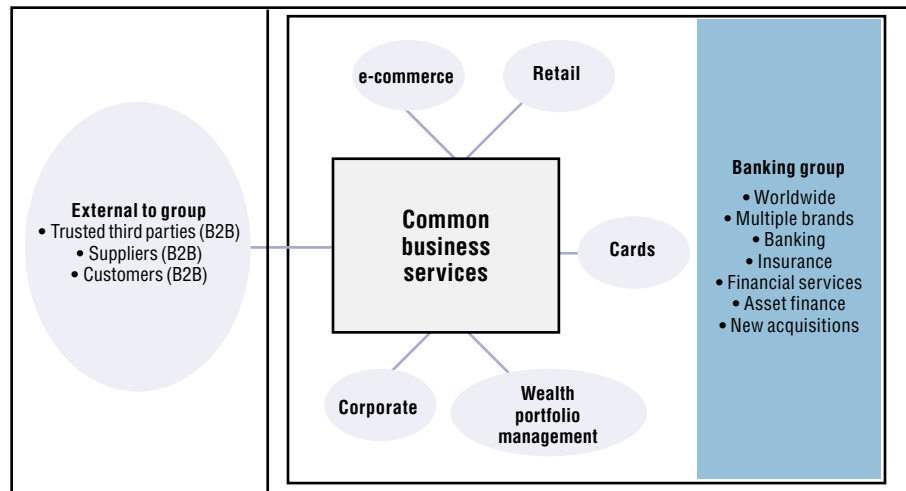


Figure 2. Common business services

Most of the common business services exist already as CICS functions that are part of the core banking applications. These functions are called today using a number of different access methods including terminal emulators and connectors, such as CICS Transaction Gateway.

Target services-based architecture

The proposed Web services infrastructure will provide business services to the group as a whole; sometimes individual brands will use the same business service, sometimes those individual brands will have specific unique requirements. The target architecture (Figure 3) allows CICS functions to be published as services and also allows CICS programs to make service calls to services outside of CICS.

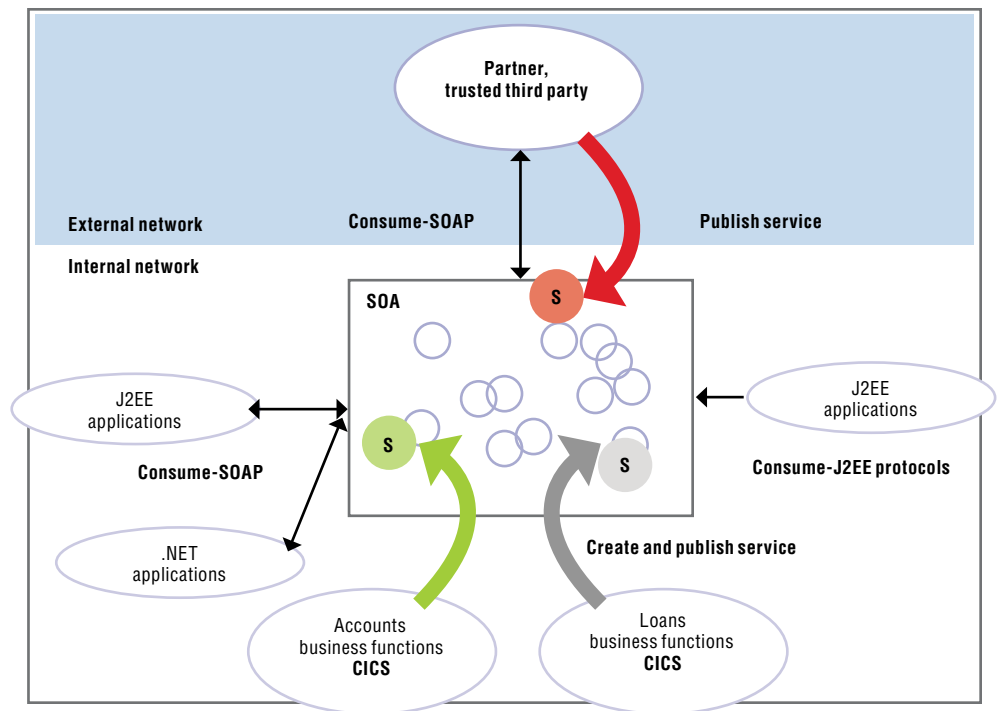


Figure 3. Target services-based architecture

Figure 3 shows how CICS services, such as accounts and loans services, are intended to be reused across the group:

- *To publish CICS services to internal group systems for consumption using J2EE protocols*
- *To publish CICS services to internal group systems for consumption using SOAP-based protocols*
- *To publish CICS services to external business partners for consumption using SOAP-based protocols*
- *To allow external business partners to publish their own services for consumption by internal CICS systems using SOAP-based protocols.*

An example of an accounts service is the retrieval of customer account information in which the service requester provides an account number and is returned account information such as customer name, address, balance and overdraft limit.

Solution requirements

The main functional requirements of the project led to the choice of using the direct exposure model, which uses CICS Web services support to enable services. The following table shows how CICS Web services support meets the major functional requirements of the project.

Functional requirement	CICS Web services support
Ability to invoke CICS Web services from any platform (including J2EE and .NET)	Web services are platform neutral.
Bidirectional support (allowing CICS to be the service provider or service requester)	CICS Web services support is bidirectional.
Support for long messages	CICS Web services support is not subject to the 32 KB limit on message length imposed by other connection options.
Ability to publish CICS services to a service registry	CICS Web services can be published to IBM WebSphere Services Registry and Repository.

An SOA deployment needs to address both functional and non-functional requirements. In *Secrets of SOA*,³ the IBM SOA Strategist Paul Dimarzio points out that the platform-neutral nature of SOA development technologies has led many people to take a matching platform-neutral view of SOA deployment. He goes on to say that SOA deployment and the target infrastructure require as much, if not more, attention than that given to development; and that larger and more complex enterprises need to take even greater care to ensure that SOA services are deployed on platforms best able to handle the job properly.

These are the main non-functional requirements and infrastructure challenges of this project:

- High availability

The business-critical Web services must be available at all times. If we plan to deploy important applications as Web services to be used both internally and externally, we need to ensure that those Web services are available in the event of planned and unplanned outages of software and hardware components.

- Security

Our solution must be secure and must meet the end-to-end security requirements of the customer. A basic security requirement is that the service requester's identity must flow with the message and that for certain high-value services (for example, a transfer request of greater than US\$5000), the target CICS transaction must run with the requester's identity.

See the following section on usage scenarios for a description of how we used a combination of traditional security mechanisms and new SOA security capabilities, such as Web services security and the WebSphere DataPower appliance, to meet the security challenge.

- Performance and scalability

The solution must meet the performance and scalability expectations of the customer. One of the prime objectives is to demonstrate how a Web services workload can be dynamically managed based on performance goals and to show that a CICS Web services infrastructure can handle the kind of workload currently supported with other access methods.

- Real-time monitoring

Problems must be automatically highlighted, and detailed real-time information (including service hit rates, response times and message lengths) must be available for problem diagnosis.

The subsequent sections of this paper focus on how the project addressed these challenges.

Usage scenarios

Generally speaking, CICS Web services can be accessed directly from a service requester or indirectly through a service bus. Using a service bus has several advantages:

- *Service requesters do not need to know the specific location of service providers.*
- *The service bus can manage the security characteristics of inbound or outbound Web services requests, change transports, and perform security tasks such as authentication, identity mapping, and asserting identities to the target service provider.*
- *The service bus can simplify the management of service deployment and configuration.*

A number of service bus implementations meet the requirements for this project. We chose to use a WebSphere DataPower Integration Appliance XI50 as a service bus component, for the following reasons:

- *There is no immediate requirement to support a large diversity of message formats and protocols (messages are XML-based and HTTPS is the transport protocol).*
- *The mediation requirements are relatively simple (routing, identity mapping, auditing and XML transformation).*
- *The DataPower appliance provides high-performance, cost-efficient processing of translation and routing instructions.*

Note that the use of the DataPower appliance as a service bus component does not rule out the use of a more centralized enterprise service bus (ESB) solution in the future (for example, using IBM WebSphere Enterprise Service Bus). These different ESB solutions can be combined into a federated ESB solution.

The following usage scenarios were tested for this project:

CICS as a service provider

Services can be accessed directly, for example, from internal service requesters running in WebSphere Application Server. These services can also be accessed indirectly through the DataPower appliance from internal .NET service requesters or external trusted third parties (see Figure 4).

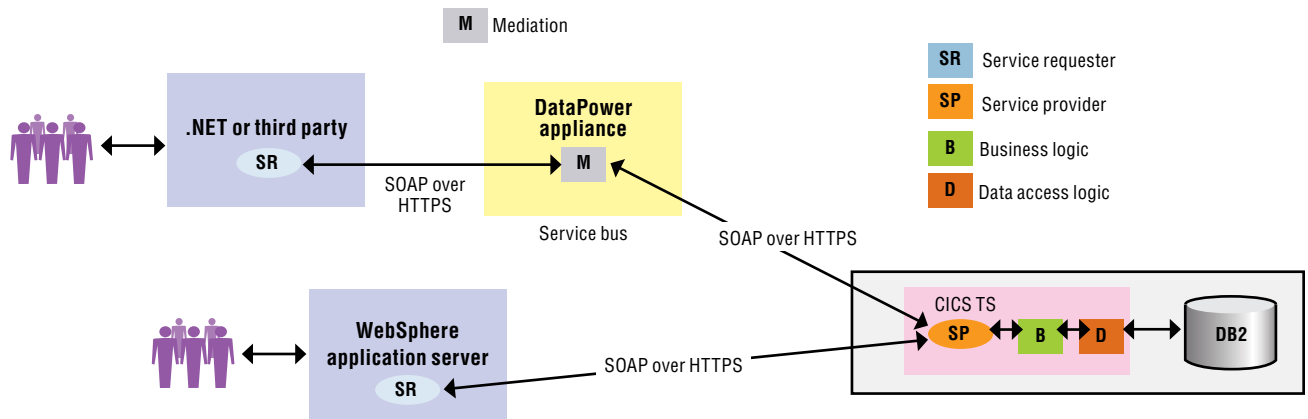


Figure 4. Scenarios for CICS as a service provider

The end-to-end security requirement is satisfied using a security solution based on identity assertion. Identity assertion is an authentication mechanism that is applied among three parties: a client, an intermediary server and a target server.

- The intermediate server (for example, WebSphere Application Server or a DataPower appliance) authenticates the client, and transfers the client's identity in a security token with the request message to the target server (CICS in this scenario). Before propagating the client's identity, the intermediate server maps the identity to an IBM Resource Access Control Facility (IBM RACF®) identity recognized by CICS, for example by calling Lightweight Directory Access Protocol (LDAP).
- The intermediate server must establish a trust relationship with CICS by authenticating itself and then by being recognized as a trusted partner of the CICS region. In this implementation, the trust relationship between the intermediate server and CICS is established using Secure Sockets Layer (SSL) client authentication.

- A custom-written CICS header-processing program, which is invoked as part of the pipeline processing for the service provider, parses the security header and assigns the RACF identity to the CICS task. The RACF identity uniquely identifies the user for which the request is being processed. CICS does the normal authorization checks, making sure that the user is authorized to use this particular service.

Identity assertion is an extended Web Services Security (WS-Security) mechanism supported by WebSphere Application Server, Version 6 and the DataPower appliance.

CICS as a service requester

When CICS is a service requester, CICS applications need access to services in WebSphere as well as to other services that will be accessed indirectly through the DataPower appliance (see Figure 5).

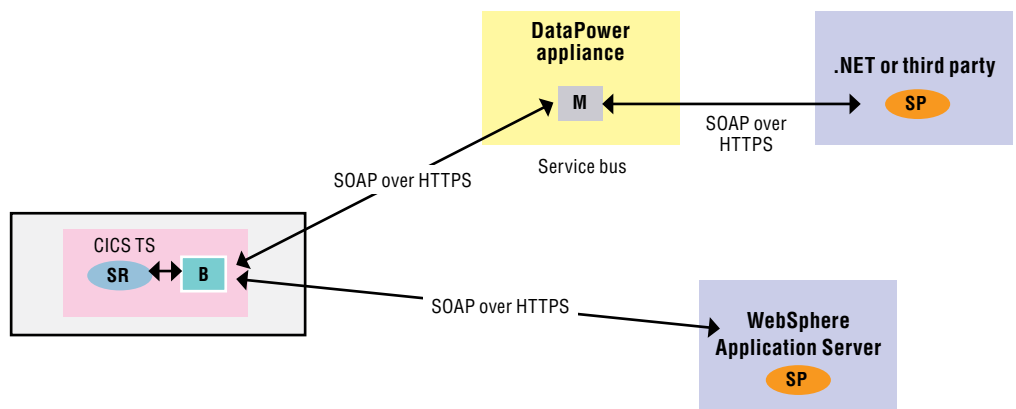


Figure 5. Scenarios for CICS as a service requester

An end-to-end security solution based on identity assertion is also used for this scenario. In this case, a custom-written CICS message handler program, which is invoked as part of the pipeline processing for the service requester, obtains the RACF user ID associated with the CICS task and sends this as a security token to the intermediate server.

Building a high-availability SOA infrastructure

Downtime is costly. The cost of downtime is so great that many of today's enterprises can no longer afford planned or unplanned outages. The statistics are staggering, as businesses can potentially face losses from tens of thousands of dollars to multiple millions of dollars per hour of downtime. Even beyond the financial aspects, downtime can also affect key areas of customer loyalty, market competitiveness and regulatory compliance. A robust configuration designed for high availability is therefore a necessity, especially in financial systems.

In this section we look at the main availability considerations for a CICS Web services implementation and outline the main availability capabilities of our configuration.

Bottom-up, top-down or meet-in-the-middle

Before enabling a CICS application as a service, you have to decide which style of Web service to use (bottom-up, top-down or meet-in-the-middle):

- *In the bottom-up approach, you generate a Web service description (WSDL file) and data mapping from a high-level data structure. You can use this method to create a service provider from an existing CICS application.*
- *In the top-down approach, you generate a data mapping and high-level data structure from a Web service description. You can use this method to create a new CICS service provider program.*
- *The meet-in-the-middle approach is similar to the top-down approach, except the CICS service provider program is a wrapper program rather than the target business-logic program. The wrapper program then links to an existing business-logic program.*

We chose the meet-in-the-middle approach because it offers the best flexibility and control over XML message formats, but it also facilitates the creation of a high-availability configuration because it allows the wrapper program to be deployed in a different CICS region from the existing business-logic programs. This means that a CICS program running in a previous CICS region can be deployed as a Web service. It also minimizes the number of CICS regions in which Web services definitions need to be installed; and it is also consistent with the principle of configuring CICS regions with different roles within a CICSplex.

For a CICS service requester, an existing business-logic program links to a wrapper program. The wrapper program uses the EXEC CICS INVOKE WEBSERVICE API to invoke a Web service.

CICSplex capabilities

For the CICS service provider scenario (Figure 6), an inbound gateway-owning region (IGOR) hosts the wrapper program. The wrapper program then links to a remote business-logic program.

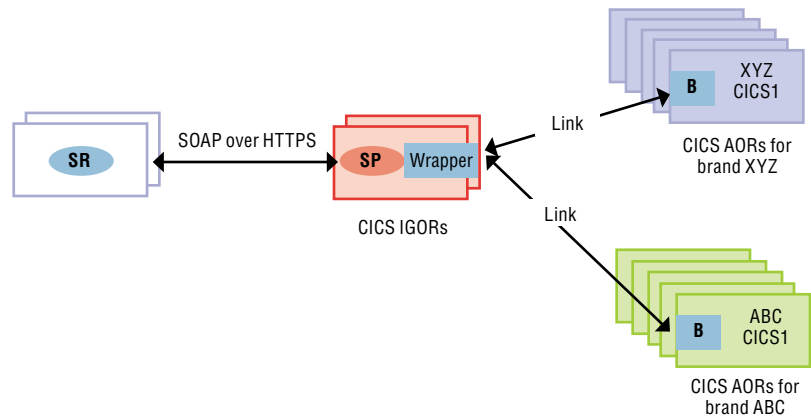


Figure 6. Inbound gateway-owning region

This customer requires the business logic associated with a particular brand within the organization to be run in a set of brand-specific application-owning regions (AORs); in our example we have a brand ABC and a brand XYZ. There is no such requirement for the IGOR, which is brand-neutral; work for different brands is processed by a single IGOR. The program link from an IGOR to a brand-specific AOR can be managed using IBM CICSplex® System Manager (CICSplex SM) workload management.

For the CICS service requester scenario (Figure 7), an outbound gateway-owning region (OGOR) hosts the wrapper program. Business-logic programs in brand-specific CICS regions do not invoke Web services directly. Instead, they link to wrapper programs in OGORs, which in turn make outbound service calls. The program links can be workload-managed across a set of brand-neutral OGORs using CICSplex SM workload management.

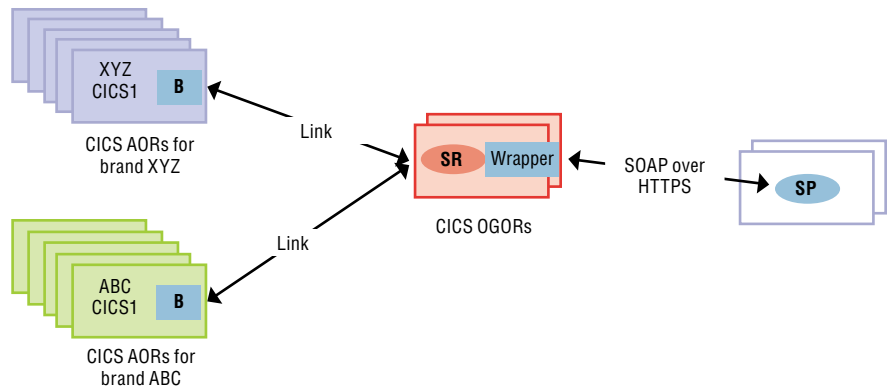


Figure 7. Outbound gateway-owning region

By providing an OGOR, you enable a CICS program running in a back-level AOR to invoke a Web service by calling a wrapper program in the OGOR.

Parallel Sysplex capabilities

When deploying CICS Web services in a Parallel Sysplex configuration, you can take advantage of workload management capabilities that are specific to the IBM z/OS® operating system, including Sysplex Distributor, TCP/IP port sharing, IBM MVS™ Workload Manager (WLM) facility and shared MVS log streams. Figure 8 shows the high-availability configuration tested for the CICS service provider scenario.

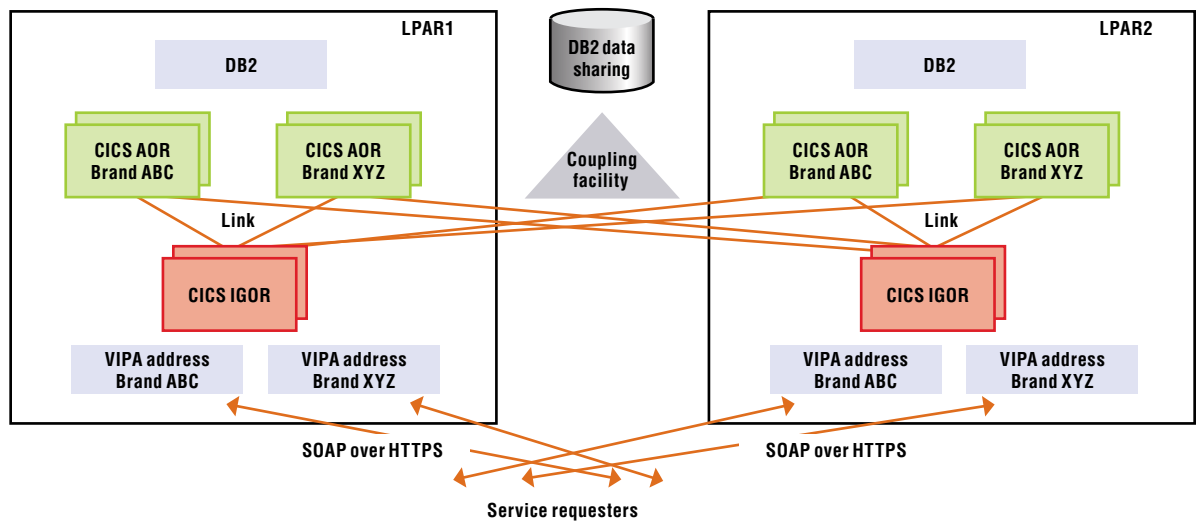


Figure 8. Parallel Sysplex configuration

- *Sysplex Distributor is used to manage workload on TCP/IP connections across two logical partitions (LPARs), LPAR1 and LPAR2.*
- *A different dynamic virtual Internet Protocol address (VIPA) and port combination is used for each brand or subsidiary of the financial group (for example, retail bank or insurance company). This establishes a virtual separation of Web services traffic across the different brands.*
- *Multiple IGORs on each LPAR listen on a shared TCP/IP port.*
- *Program link requests are dynamically routed to cloned brand-specific AORs using CICSplex SM.*
- *A user log stream is shared in the coupling facility so that all IGORs write audit records to the same journal (audit records are written for each high-value service request). The audit records are written by a custom-written CICS header-processing program.*
- *CICS business-logic programs running in the AORs share access to business data using data sharing provided by IBM DB2® software.*

Validating the solution

Before deploying a new IT solution, it is important to validate the solution with a set of tests designed to prove that the solution meets the non-functional requirements. In this section we provide some examples of the tests that were conducted to validate the solution.

Avoiding service outages

The objective of the service availability tests is to demonstrate continuous availability of the CICS Web services infrastructure, for both planned and unplanned outages. Based on the configuration shown in Figure 8, IBM Rational® Performance Tester¹ was used to inject a Web services workload. After reaching a steady service hit rate, a number of failure scenarios were tested.

Figure 9 (from Rational Performance Tester) shows the service hit rate and user load for Account inquiry service requests.

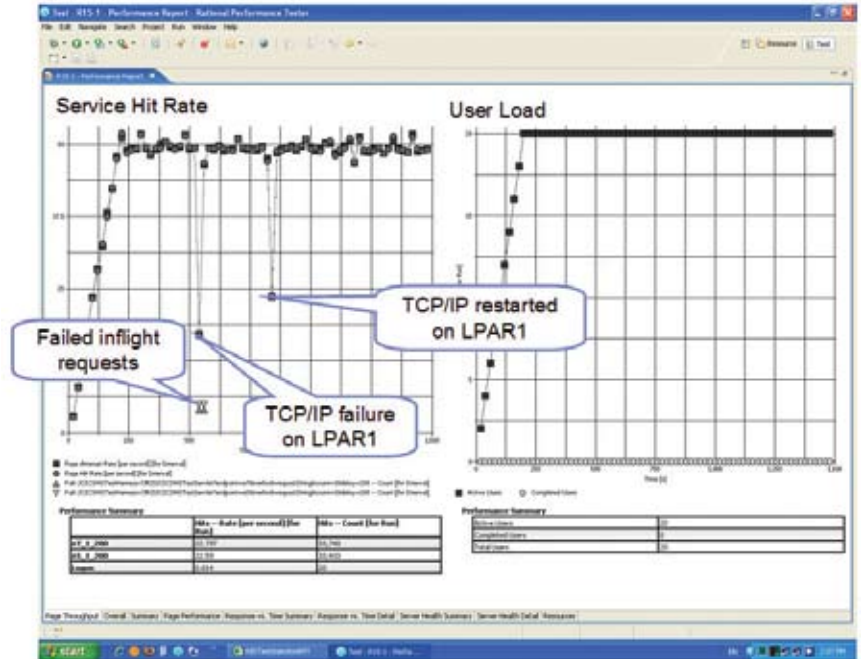


Figure 9. High-availability test

- A workload is simulated with 20 Web service clients such that a steady state service hit rate of 50 services per second is achieved.
- TCP/IP is cancelled on LPAR1 (the primary stack for Sysplex Distributor). We can see that there is a brief fall in service hit rate, and a corresponding increase in service response time, at the time when TCP/IP is stopped. The service hit rate then returns to the same level as before the failure. At the time of the failure, all active TCP/IP connections to LPAR1 are disconnected and all new connection requests are routed to LPAR2.
- When TCP/IP is restarted on LPAR1, it takes back ownership of the VIPA, and we see another brief, but less marked deterioration in service hit rate.

This test demonstrates that the user perceives the service as being continuously available throughout the scenario, albeit with increases in service response time at the time of the failure and with a very small number of failed requests corresponding to those that were “in flight” at the time of the TCP/IP failure.

Throughout the failure scenario testing, the Web user interface (WUI) of CICSplex SM was used to manage the CICSplex. The CICSplex SM WUI can be used to monitor a CICS Web services workload, including the number of active TCP/IP connections and the number of times a Web service is used. For example, Figure 10 (from the CICSplex SM WUI) shows that the Account inquiry service AcntInq has been called 14071 times on CICS system CICSIG01, and 9147 times on CICS system CICSIG02.

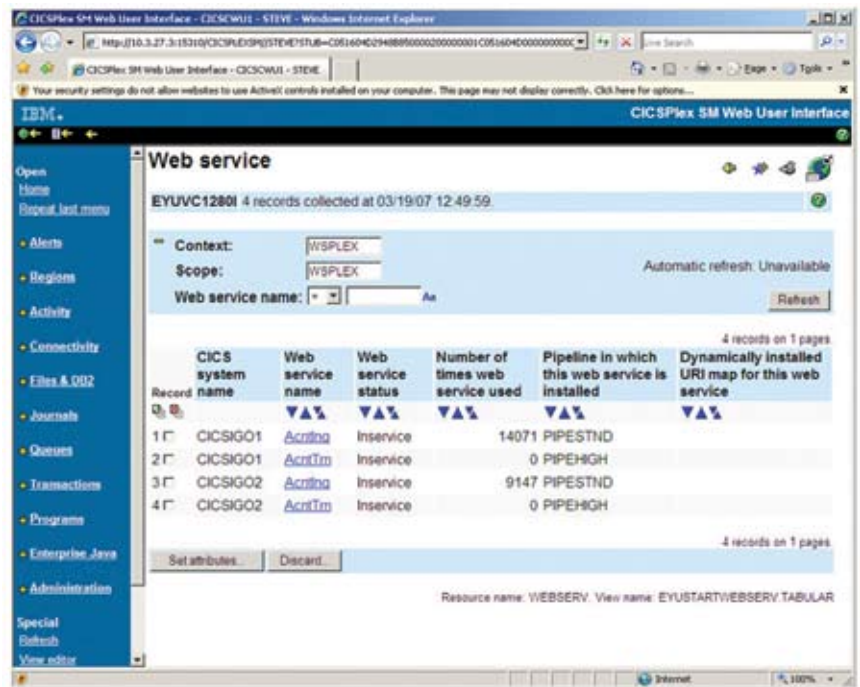


Figure 10. Monitoring CICS Web services with CICSplex SM

Performance and scalability

This section outlines the performance tests that we conducted to prove the scalability and responsiveness of our SOA implementation. The objective of our performance testing was to validate that the solution could meet the customer’s performance requirements, in terms of scalability, response times and processor time.

We defined a set of scalability and performance tests to validate the solution:

- *Gradually increasing the number of simulated Web-service invocations to test for linear scalability*
- *Investigating the performance impact of varying SOAP message lengths*
- *Measuring the performance cost of different Web-service security mechanisms*

Linear scalability

Linear scalability implies that for a single service request, the amount of processor power required, and the time taken to process the request, remain relatively constant as the number of concurrent service requests being processed increases. The amount of processor power required is measured by the number of milliseconds of processor (CP) time taken by the IGOR to process a single request. Figure 11 shows how many milliseconds of processor time were required to process an Account inquiry service request for different numbers of concurrent service requests.

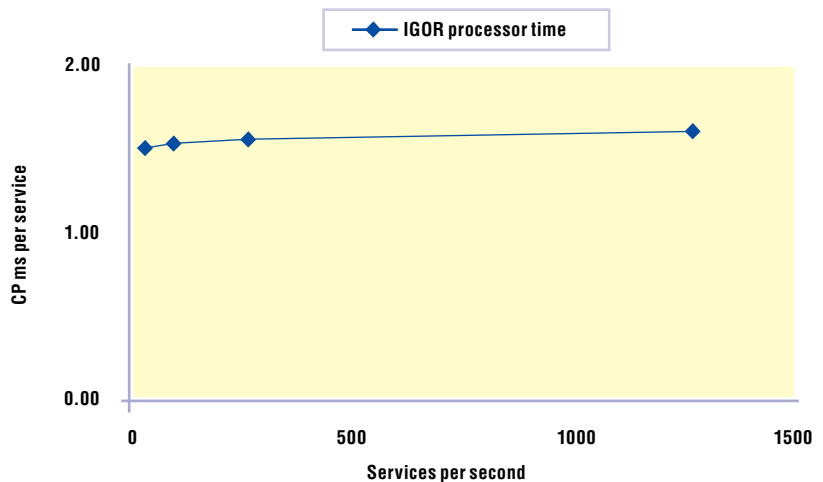


Figure 11. Processor time required by the Account inquiry service

Figure 11 shows that the milliseconds of processor time required to process an Account inquiry request is relatively constant, thus demonstrating linear scalability. The response time also demonstrated linear scalability by remaining constant for tests at different service rates.

SOAP message length

We performed tests to evaluate the effect of message size on the performance of the Account inquiry Web service. We varied the message size by increasing the number of XML elements contained in the response message returned by CICS.

Figure 12 shows the effect of message size on processor time required for the Account inquiry service.

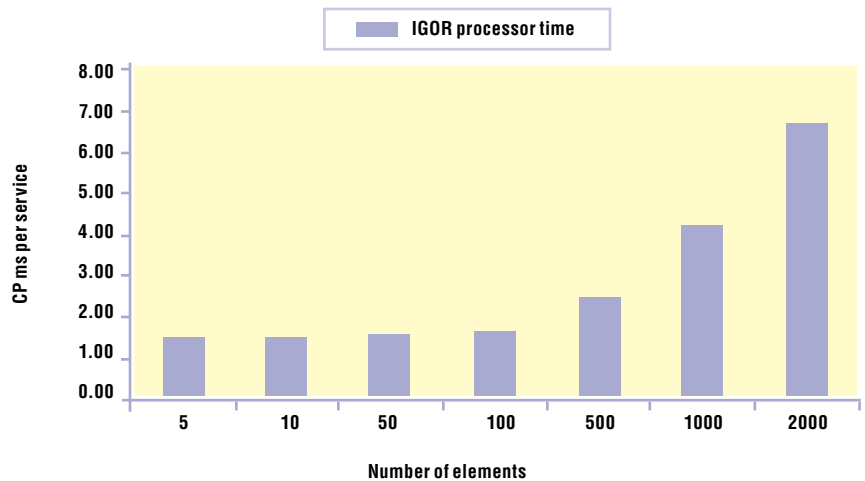


Figure 12. Effect of SOAP message length on processor time

Figure 12 shows that the milliseconds of processor time required to process an Account inquiry request depends on the number of elements returned in the response message. In our test, a SOAP message of 50 elements contains 1 KB of business data, but the message itself has a length of 2.6 KB because of the XML tags. The longest message is 400 times larger than the smallest message, but the increase in processor time is less than fivefold.

Figure 13 shows the effect of message size on response time for the Account inquiry service.

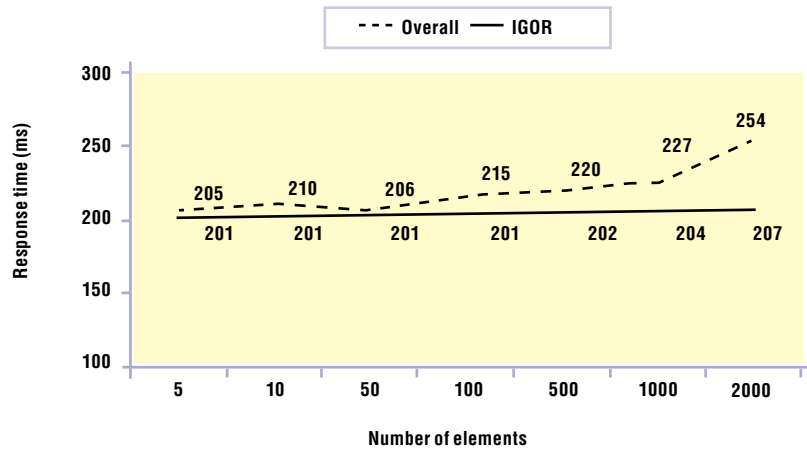


Figure 13. Effect of SOAP message length on response time

Figure 13 shows the IGOR response time and overall response time for Account inquiry services with varying response message sizes.

- The IGOR response time is the average response time recorded by the IBM Resource Measurement Facility (IBM RMF™) component of z/OS for the CICS service-provider transaction. The response time ranges from 201 milliseconds for the smallest message length, to 207 milliseconds for the largest message length. Note that for this test, a 200 ms delay is specifically programmed into the target business-logic program in order to simulate the average response time for existing programs.
- The overall response time is the response time measured in the workload simulation tool, and so includes the response time of the CICS service-provider application and the service-requester application (a J2EE application running in WebSphere Application Server).

We see that the IGOR response time remains relatively constant while the overall response time increases more significantly for the longer SOAP messages. This is because CICS is building the SOAP message response (no additional XML parsing is required), whereas WebSphere Application Server is parsing the response message. Note that the complexity of the SOAP message, such as the number of elements and message structure, also affects performance.

Security cost

We performed a test to evaluate the effect of different security models. Figure 14 compares the processor time required for two services that use different security models, Account transfer and Account inquiry. For Account transfer requests, a user-written CICS header-processing program is invoked as part of the pipeline processing for the service provider. The header-processing program parses the security header, writes an audit record and assigns the caller's RACF identity to the CICS task. For Account inquiry requests, the CICS task does not run with the caller's identity.

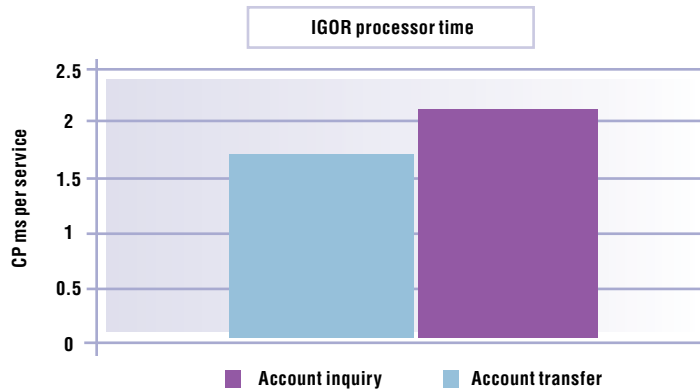


Figure 14. Effect of security model on processor time

Figure 14 shows that the processor time required for the Account transfer service is approximately 20 percent greater than that of the Account inquiry service. This is due to the overhead of processing the security header, in particular, the additional cost of switching the security context to the caller's identity.

Real-time monitoring

To meet the monitoring and systems-management requirements, the solution needed to perform these functions:

- *Monitor against a set of predefined goals for service performance*
- *Disable a Web service and quiesce an IGOR or OGOR*
- *Withstand failures of CICS systems*
- *Prevent a problem with one brand from affecting the service level of other brands*
- *Identify a problem when it occurs and identify the location and root cause of the problem*

The following tools were used to monitor the CICS Web-services infrastructure:

- *CICSplex SM WUI for operations, management and monitoring*
- *IBM Tivoli OMEGAMON® XE for CICS for more detailed analysis of Web services in CICS, including tracking against service response-time goals, and integration with the IBM Tivoli Enterprise™ Portal Server (TEPS)*
- *IBM Tivoli Composite Application Manager (ITCAM) for SOA to monitor Web services across different runtime environments, including CICS and DataPower*

Figure 15 shows how the Tivoli OMEGAMON XE for CICS and Tivoli Composite Application Manager for SOA tools were used to graphically display information in a single, integrated console (Tivoli Enterprise Portal).

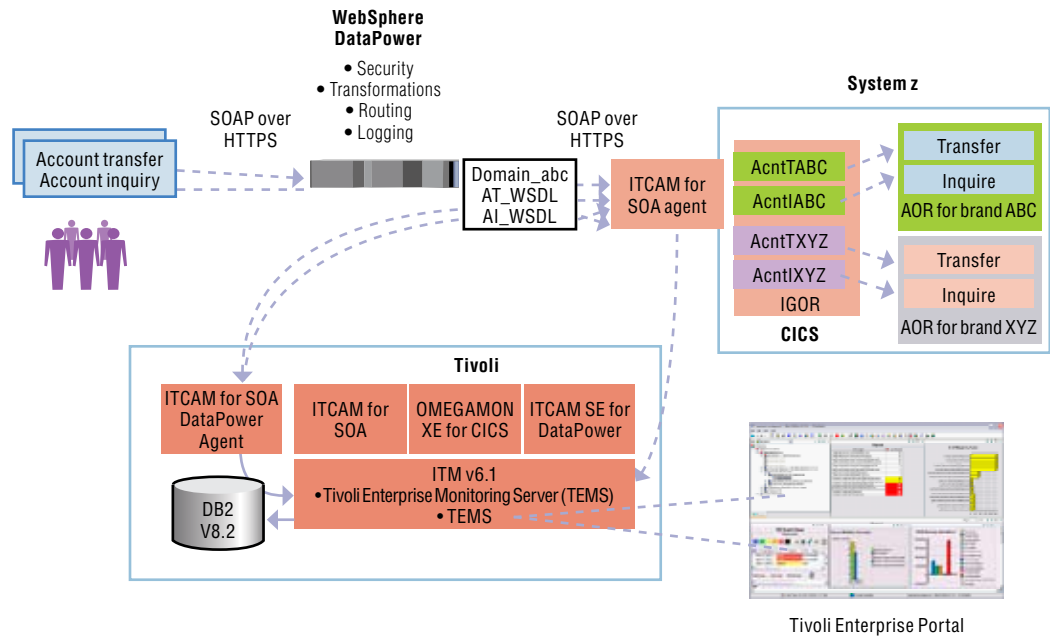


Figure 15. Tivoli monitoring configuration

To monitor all the service platforms, Tivoli Composite Application Manager for SOA employs (and includes) a series of agents such as the CICS agent, which is implemented as a message handler, and the DataPower agent, which resides on a separate machine and queries the DataPower appliance using the DataPower SOAP interface.

Tivoli OMEGAMON XE for CICS provides detailed information, such as URIMAP analysis. A URIMAP is a CICS resource definition that is used to map the Uniform Resource Identifiers (URIs) used by Web service clients to specific Web services and runtime characteristics for those services (such as a transaction identifier). Figure 15 shows the four URIMAPs used in our tests that map account inquiry and account transfer requests for both brands ABC and XYZ.

After a specific transaction identifier has been associated to a Web service request, you can use service level analysis to set response time goals for different service level classes. Figure 16 (from Tivoli OMEGAMON XE for CICS) shows how we monitored our Web services against the response time goal (1 second). We used the default service class naming, which simply groups transactions together based on the first letter of the transaction identifier. In our case, the ITRANS group represents inquiry transactions and the TTRANS group represents transfer transactions. More-specific service class names can be created based on other criteria, for example, by brand and service request type.

Figure 16 shows that for a particular test, the Web-service response time for the ITRANS group is 0.2 seconds, and the response time for the TTRANS group is 0.35 seconds.

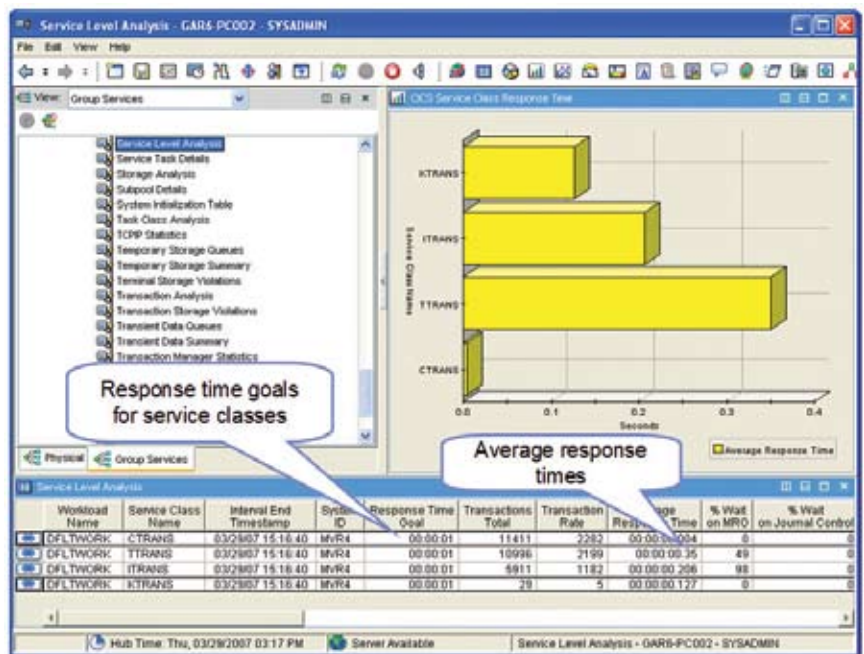


Figure 16. Service level analysis with Tivoli OMEGAMON XE for CICS

Whereas OMEGAMON XE for CICS provides the CICS view of Web service health, Tivoli Composite Application Manager for SOA is designed to monitor Web services across the different platforms of an SOA environment. These runtime environments include WebSphere Application Server, CICS and the WebSphere DataPower Integration Appliance XI50. The primary advantage of this cross-platform view is that it helps IT operators to identify the source of problems across the IT infrastructure from a single console.

Figure 17 shows how, in addition to service response time, Tivoli Composite Application Manager for SOA also provides information about message counts and message sizes for each operation of a CICS Web service.

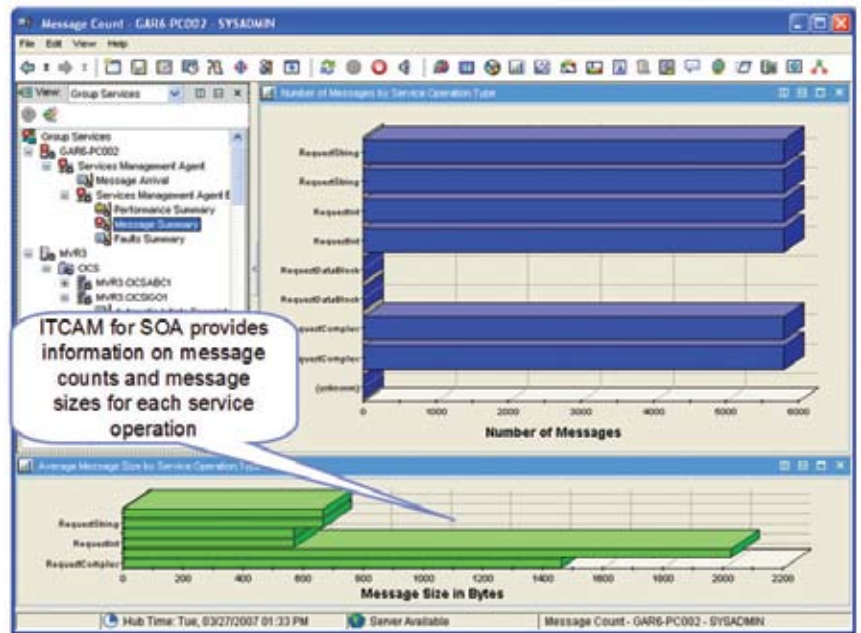


Figure 17. Message analysis with Tivoli Composite Application Manager for SOA

Similar information can be retrieved from WebSphere Application Server and the DataPower appliance, providing the operator with a system-wide view of the services.

Conclusion

This paper describes a CICS Web-services configuration based on the non-functional requirements of a particular customer:

- *A combination of traditional security mechanisms and new SOA security capabilities, such as Web services security and the WebSphere DataPower appliance, are used to meet the security requirements.*
- *CICSplex SM and specific z/OS workload-management capabilities, such as Sysplex Distributor, TCP/IP port sharing and the MVS Workload Manager (WLM) facility are used to create a high availability infrastructure.*
- *IBM Tivoli monitoring products are used to monitor the Web services infrastructure.*

This example demonstrates that it is possible to embrace important strategic initiatives such as SOA, and at the same time continue to derive value from tried and trusted CICS applications.

For more information

To learn more about CICS Web services, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/cics



© Copyright IBM Corporation 2007

IBM United Kingdom Limited
Hursley Park
Winchester
Hampshire
SO21 2JN
United Kingdom

Produced in the United States of America
12-07
All Rights Reserved

IBM, the IBM logo, CICS, CICSplex, DataPower, DB2, MVS, OMEGAMON, Parallel Sysplex, RACF, Rational, RMF, System z, Tivoli, Tivoli Enterprise, WebSphere and z/OS are trademarks of International Business Machines Corporation in the United States, other countries or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

¹ Rational Performance Tester is a multi-user load-testing and performance-testing tool for validating the scalability of Web applications.

² The IBM Design Centers for on demand business are state-of-the-art facilities where certified IT architects and specialists work with clients from around the world to design, architect and prototype advanced IT infrastructure solutions to meet the challenges of on demand computing. To learn more about the Design Centers, visit ibm.com/systems/services/designcenter/.

³ DiMarzio, Paul, SOA Strategist, IBM, et al. *Secrets of SOA*. Larstan Publishing, Inc. This document provides an enterprise view on SOA deployment. To read more, visit www.secretsofsoa.com/.