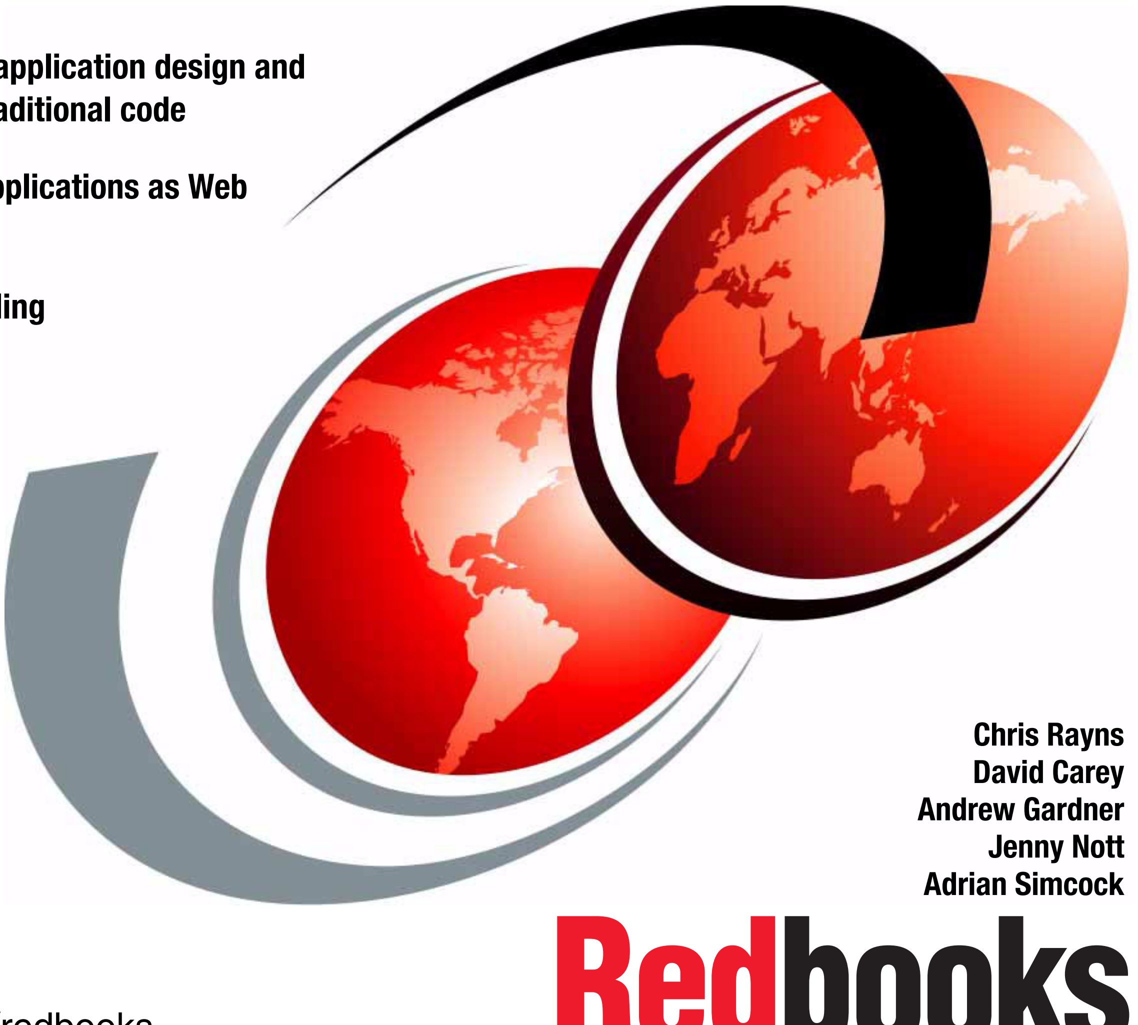
IBM

Developing Web Services Using CICS, WMQ, and WMB

Bottom-up application design and re-use of traditional code

Exposing applications as Web services

Modern tooling techniques



Chris Rayns
David Carey
Andrew Gardner
Jenny Nott
Adrian Simcock

Redbooks

ibm.com/redbooks

International Technical Support Organization

Developing Web Services Using CICS, WMQ, and WMB

September 2007

SG24-7425-00

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

First Edition (September 2007)
This edition applies to Version 3, Release 1, CICS Transaction Server.

Contents

Notices

Trademarks

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

AIX®	Language Environment®	SupportPac™
C/370™	MQSeries®	System z™
CICS®	MVS™	TXSeries®
CICSPlex®	Rational®	WebSphere®
DB2®	Redbooks®	z/OS®
IBM®	Redbooks (logo) ®	zSeries®
IMS™	REXX™	

The following terms are trademarks of other companies:

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

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

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

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

Preface

This IBM® Redbooks® publication provides a practical demonstration of how to develop applications that take advantage of CICS® Web services facilities. This book can be viewed as a follow-on from the IBM Redbooks publication *Application Development for CICS Web Services*, SG24-7126-00, with the addition of using modern tooling techniques. Because we are creating a new application, we follow the bottom-up approach described in *Application Development for CICS Web Services*, SG24-7126-00. Although not a requirement, we highly recommend that you review that publication for a much deeper discussion of CICS Web services development topics and alternative approaches.

The primary purpose of this book is to demonstrate that well structured CICS Web services are easy to develop using the CICS Web services assistant. We also take a look at modern tooling such as WebSphere® Developer for zSeries® (WD/z), which the traditional mainframe developer may not have seen before. As traditional developers we found the Eclipse-based interface to the mainframe a new and exciting way to both interact and use facilities on the mainframe.

In the Redbooks publication *Application Development for CICS Web Services*, SG24-7126-00 we showed how to expose a CICS application, namely the CICS Catalog Manager sample application. We now take this process one step further by developing a CICS application from the ground up that will be CICS Web Services enabled. We initially develop this application that runs in CICS using standard BMS (3270) panels as an initial front-end. As the application and display logic are separate, we are well placed to use the business functions as CICS Web Services creating the WSDL interface using the CICS Web services assistant.

The team that wrote this Redbooks publication

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

David Carey is a Senior IT Advisory Specialist with the IBM Support Center in Sydney, Australia, where he provides defect and non-defect support for CICS, CICSPlex/SM, the WebSphere MQ family of products, and z/OS®. David has worked in the IT industry for 27 years and has written extensively about CICS and zOS Diagnostic procedures for the ITSO.

Andrew Gardner is an IT Availability Specialist with the IBM Support Centre in Canberra Australia, where he provides specialist support in the WMQ, Message Broker and related WBI products. Andrew holds a science degree from the University of Sydney in Computer Science and Pure Mathematics. He has worked for over 20 years in the IT industry, primarily on mainframe systems, specializing in CICS and WMQ.

Jenny Nott is an IT Availability Specialist with the IBM Support Centre in Sydney, Australia, where she provides defect and non-defect support for CICS, CICSPlex® SM, MQSeries®, and TXSeries®. Jennifer has 16 years of experience in Information Technology, and holds an honors degree in Applied Science from the University of NSW, majoring in Physical Geography.

Adrian Simcock is a Software Engineer working at the IBM Australian Programming Centre in Perth, Australia. He has 20 years of experience working with CICS and related products for a variety of IBM internal and external customer accounts. He is currently working as a developer for IBM Fault Analyzer. He holds a degree in Electrical and Electronic Engineering from Plymouth University.

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

Bob Haimowitz,
Rich Conway
International Technical Support Organization, Poughkeepsie Center

Lyn Elkins
IBM US

and others who kindly reviewed this IBM Redbooks publication and offered excellent comments.

Become a published author

Join us for a two-to-six week residency program! Help write an IBM Redbooks publication 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 the following Web site:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other 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 email 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

In this chapter we discuss the reasons why a large CICS site would consider using CICS Web Services as an interface to their heritage application systems. We also give a brief overview of CICS Web Services and introduce the application we develop in this book.

1.1 Why we wrote this book

This book is intended to be of most use to the CICS Application Developer / Architect who needs to see a practical demonstration of how to develop applications that take advantage of CICS Web services facilities. This book can be viewed as a follow-on from the IBM Redbooks publication *Application Development for CICS Web Services*, SG24-7126-00, with the addition of using modern tooling techniques. Because we are creating a new application, we follow the bottom-up approach described in *Application Development for CICS Web Services*, SG24-7126-00. Although not a requirement, we highly recommend that you review that publication for a much deeper discussion of CICS Web services development topics and alternative approaches.

The primary purpose of this book is to demonstrate that well structured CICS Web Services are easy to develop using the CICS Web services assistant. We also take a look at modern tooling such as WebSphere Developer for zSeries (WD/z) which the traditional mainframe developer may not have seen before. As traditional developers we found the Eclipse-based interface to the mainframe a new and exciting way to both interact and use facilities on the mainframe.

In the Redbooks publication *Application Development for CICS Web Services*, SG24-7126-00 we showed how to expose a CICS application, namely the CICS Catalog Manager sample application. We now take this process one step further by developing a CICS application from the ground up that will be CICS Web Services enabled. We initially develop this application that runs in CICS using standard BMS (3270) panels as an initial front-end. As the application and display logic are separate, we are well placed to use the business functions as CICS Web Services creating the WSDL interface using the CICS Web services assistant.

The application we develop is a simple but new concept. It uses the publish / subscribe functionality of WebSphere Message Broker (WMB) and therefore WMQ as a backbone. External subscribers, on receipt of a notification will invoke a sequence of CICS Web Services. This application is described in 1.5, “The Change of Address application” on page 8 and the technical implementation details are fully described in Chapter 5, “Development of the Change of Address CICS application” on page 73.

Above all else, we want to demonstrate that creating Web services from new and existing CICS applications using the CICS Web services assistant is easy and worthwhile.

Of course, one of the chief concerns when exposing CICS functions as Web services is security. This is a topic that is covered at length in another Redbooks publication *Securing Access to CICS Within an SOA,* SG24-5756-01.

1.2 Why use CICS Web services

A great many large commercial institutions, particularly in the financial sector have been using CICS Transaction Server for decades now. The reasons are many: robustness, performance, scalability, security, and so forth. The basic transaction processing heart of CICS is still the same at the API interface, but under the covers, many new features and performance improvements are introduced. The choice to use these new features is left entirely as a business decision.

IBM introduced support for a number of new interfaces for technologies that did not exist when CICS was developed. The CICS Web Interface with TCP/IP support is one such interface. More recently, support for CICS Web services has become formally available with CICS TS V3.1. This is a very significant development for heritage CICS sites as it potentially puts their CICS applications on an equal footing with the most recently developed and distributed Web applications. This is because Web Services are a standard, platform-independent method of invoking a remote function. The caller has no need to know the implementation details of the target service.

However new facilities in CICS are not always keenly embraced by these long-term CICS users. The basic CICS functionality remains the same. Indeed there are many applications, typically CICS COBOL, that have been running largely unchanged for 20 years or more. As time moves on and applications support skills move on, there is understandable apprehension to change working code to employ new functionality unless there are significant justifications to do so. With Web Services, there is ample justification to take the plunge and if the application is well designed, minimal changes are required to the heritage base code.

The following sections summarize the reasons that a long term CICS site should seriously consider adapting a large, working CICS application to enable CICS Web Services.

Competitive advantage

By having parts of existing corporate applications available outside the internal company intranet, clients and suppliers are enabled for electronic commerce. This is certainly not a new concept, but EDI solutions have always required intermediate bridging software to transform and route requests and responses. Such software was usually proprietary and required at both ends of the connection as well as being expensive for smaller sites.

Using a standard interface

By using the Web services approach, a standard interface is published enabling clients and suppliers to develop their Web service clients in their own time, using their own tooling, independently of the target software or platform. The quicker and easier it is for clients and suppliers to connect to the corporate Web service, the more likely they will use the Web service.

Cost advantage

There are still a great many heritage applications running on mainframes notably using CICS (3270, Client-Server, Web, Bridge, and so on). The cost and scale of redevelopment of these systems is overwhelming to many organizations. Enabling these established systems for Web services is highly desirable for these sites.

Employing a service oriented approach does not necessarily mean a huge outlay for new hardware and software. Indeed the aim of this book is to demonstrate how new and existing CICS applications can be adapted as Web services without major outlays. CICS provides utilities that enable existing CICS applications to be adapted as Web services. At the simplest level, this is all that is required to enable a CICS program to become a Web service. In practice however, issues such as complex data types and security must be taken into consideration.

Centralized infrastructure

Keeping the Web services infrastructure on the mainframe has a number of important features that will appeal to sites with a long and rich association with CICS. From those sites, issues such as centralized security, backup, and administration are fundamentally important to their business requirements.

Typically business resumption strategies are in place, tested, and they have no desire to change these for the sake of a few application niceties. None of these key features need be affected by employing CICS Web services.

A modern approach to e-commerce

From the view of many businesses, it is highly desirable to use modern application development approaches. This not only allows them to take advantage of new technologies but can relieve their mainframe of some the development processing burden. It is also important as these businesses need to be able to attract quality graduate programming staff. Potential new recruits will be much more interested working for a business that uses modern techniques as they frequently view roles, such as application maintenance, (rightly or wrongly), as being minimal-interest jobs.

Long term staff also need to keep their skills up-to-date. They will not fall behind as a result of the modern mainframe infrastructure. Indeed the interconnection of heritage and new technology can be a very interesting challenge.

Corporate image

The use of new technology is also very much part of a corporate image. Companies who are slow to react to IT trends can be viewed as being slow off the mark. This may or may not be correct, but perception is important. At the most superficial level, use of CICS Web Services can provide a modern make-over to heritage application functions for limited investment.

Timely response to business needs

There is an increasing desire to be positioned for quick response to business needs. By using standard, secure, and well accepted interfaces, requirements such as EDI, or even business integrations (see below) become significantly easier. This can be done using applications infrastructure such as messaging, but there are many, possible incompatible messaging systems available. While this might be practical within an organization, cross organizational messaging can become a problem. Web services are transport independent. The most widely used invocation method for Web services is SOAP over HTTP, but it is just as valid to use SOAP over WebSphere MQ links or JMS.

Corporate acquisition driven integration

Just about every corporation undergoes acquisitions or restructuring at some point these days. This usually has significant implications for the IT infrastructure of both organizations. By using a standard CICS Web Services interface, the pain of integrating such systems can be reduced. The difficulty of such integrations can never be entirely removed, which is why there is a whole industry dedicated to software integration platforms.

Software such as WMQ, WBI Adapters, and WebSphere Message Broker are significant parts of the IBM software integration portfolio. However, with some forethought, employing standard interfaces such as CICS Web Services, transitions between corporate states becomes much more manageable.

Desire to retain mainframe infrastructure

The benefits of centralized security, databases, and general administration are clear to long-term mainframe users. Some corporations have long histories (decades) with IBM mainframes. Financial and government institutions typically fall into this category, as well as those organizations where very high levels of security have always been required, such as defence or R&D. For these types of organizations, the benefits of centralized mainframe security are well established. By introducing CICS Web Services, centralized security can be retained on the mainframe. Web services allow for the inclusion of security information for authentication and authorization purposes.

1.3 Application Development in CICS TS3.1

As previously mentioned, the full support of CICS Web services was realized in CICS TS V3.1. In this section, we list and briefly discuss the functional enhancements that are relevant to Web services. This information is of most interest to the application programmer to help them forge their way into new technologies with advanced capabilities and support for modern programming techniques.

1. **Access to CICS**
   - Web services support
   - Enhanced HTTP support
   - Improved SSL support
   - Support for mixed case passwords
   - Improved user ID checks for the START API command
2. **Application transformation**
   - Enhanced C/C++ support
   - Enhanced Open Transaction Environment
   - Language Environment® MAIN support for Assembler
   - Enhanced inter-program data transfer
   - Threadsafe Web API commands
   - 64 bit addressing toleration
   - Code page conversion enhancements
   - Information Centre on an Eclipse based platform

While all of these enhancements are covered in extensive detail in the CICS TS 3.1 Release Guide, those of most interest to this book are described in further detail.

1.3.1 Access to CICS

CICS TS V3.1 includes a range of new and improved capabilities that enhance access to CICS. Standard interface and communication protocols mean that you have the facilities to exploit new technologies and re-use your CICS applications within a flexible operating environment. The potential benefits of this were discussed in section 1.2, “Why use CICS Web services” on page 3 and includes simplified development processes, reduced development costs, and reduced time to deployment.

CICS TS V3.1 delivers major new support for Web services, which is an evolution of the functions previously provided as the SOAP for CICS optional feature. These enhancements allow CICS-based applications to be exposed as Web services, thus enabling existing applications to be integrated within a service-oriented architecture (SOA). Chapter 2 develops the discussion regarding the SOA terminologies and how these relate to CICS TS and Web services.

There is also support for the WS-Atomic Transaction specification enabling distributed transaction coordination for participating partners complying with this standard. A message-level security function that complies with the WS-security specification was been provided in CICS TS V3.1. Although there is no demonstration of this feature in this publication, there is more information about this in Chapter 2, “Service-oriented architecture and CICS” on page 11 for reference.

1.3.2 Application transformation

This second group of enhancements allow existing applications to be further developed and new applications to be constructed using contemporary programming languages, constructs, and tools. Support is introduced for totally Language Environment enabled Assembler application programs. All the EXEC CICS Web API commands were made threadsafe, and there is a more efficient use of z/OS multiprocessor capabilities through enabling of the Open Transaction Environment (OTE) support to use open TCBs.

One of the significant enhancements for the application developer is the new mechanism for inter-program data transfer, using constructs called channels and containers. These provide an alternative to COMMAREAs and are not subject to the same 32 KB size restriction. This is discussed further in 2.7, “Channels and containers” on page 25.

1.4 WebSphere Message Broker and WMQ

We use some other significant software components in this project. In this section we briefly describe these components.

1.4.1 WebSphere MQ (WMQ)

WebSphere MQ (WMQ or MQ) is data transportation middleware, and has become the messaging standard for most business sectors. It assures once only delivery of messages (data records) on 30+ operating systems, using a consistent API for a wide range of supported programming languages. The language support ranges from z/OS Assembler to .NET to JMS. WMQ is available on z/OS and 30+ distributed platforms. On most distributed platforms there are two offerings—a freely downloadable client and a server (queue manager). WMQ is an asynchronous transport model, though many applications use it in a synchronous fashion.

WMQ supports the most popular network communication protocols, with the vast majority of message traffic flowing across TCP/IP networks. It also provides a simple codepage data translation, most useful when the messages are in printable character format.

WMQ supports two messaging styles, persistent and non-persistent. Persistent messages are recoverable, from logs, in the event of a queue manager outage (planned or unplanned). Non-persistent messages are not logged and are not usually recovered in the event of a queue manager outage. Non-persistent messages are faster and less expensive than persistent, but they are not assured.

The WebSphere Message Broker (WMB), which we discuss in the next section, is based on WMQ.

1.4.2 WebSphere Message Broker (WMB)

WMB provides a number of significant functions on top of WMQ. Message data can be mapped to known structures, modified, augmented, enriched, and distributed. WMB provides a development toolkit that allows visual development of message flows and data structures. Although WMB uses WMQ as a foundation, it is not limited to using WMQ as a transport mechanism. A comprehensive range of protocols are supported.

1.5 The Change of Address application

The application we shall develop to demonstrate the use of CICS Web services is quite simple, yet has a number of features that makes it ideally suited as a CICS Web service application. This application introduces the idea of a centralized repository of names and addresses, to be kept, ideally by a nation’s postal service. It keeps no personal data other than names and addresses. The purpose of this application is to notify subscribers when a person or business changes address. Subscribers would be banks, utility companies, government agencies and so on, that is, anyone who needs to know. Subscription control is left with the postal service. Address change notifications are driven by the person changing address, the *relocatee*, either by personally notifying the postal service or possibly via a Web interface, which is not included as part of this project. Of course adequate identification and proof of address are required.

The sample application safely assumes all subscriber organizations maintain a database of their clients. We provide a Web service called GetHash that returns a hash value for a given address. The organization should add a column to their customer database table to include the hash value. The generated hash value depends on a standard format of address. We provide a Web service called StandardAddress that takes an address and returns the address in a standard format. This allows for correct comparison of address hash-values.

When an address is updated in the postal service system a notification is generated and distributed via the publication/subscription function of WebSphere Message Broker. Only those corporations subscribing to the service will receive such notifications via their WMQ clients. These notifications are very small, just a hash of the old address for the relocatee. The corporation must check to see if they have a match for this hash code in their client database. If they do, they invoke a CICS Web service called RetrieveAddress that effectively asks if the address change was for their client. The CICS Web services will either reply yes and also provide the new address or no. It is then up to the corporation to update their database. The CICS RetrieveAddress Web service will cut an audit record to indicate that a particular corporation requested the new address and presumably updated their database.

This last point is important for privacy legislation reasons in many countries. The postal service may have to produce a report listing those organizations to which it has provided this information. We created a Web service called CorpAck that allows the listing of corporations that acknowledged the change of address. This is also important to the relocatee, as they otherwise have no visibility of who has or has not been notified.

We chose to implement this system in CICS because the address database is potentially a very large database or possibly even a VSAM file. There is also a possibility that many postal services may already have databases with this information. We need high performance from our CICS Web services to make the facility attractive to use. On any given day potentially hundreds or even thousands of relocations may be recorded.

2

Service-oriented architecture and CICS

This chapter introduces service-oriented architecture (SOA) and how it relates to CICS Transaction Server on IBM z/OS platform. Following this, we outline the Web Services technologies, Web services support in CICS, and how you can transform existing CICS assets to play a role in the SOA solutions.

2.1 An introduction to SOA

There is an increasing demand for technologies to support the connecting or sharing of resources and data in a flexible and standardized way, which becomes a challenge when there are different implementations of this technology both across and within company boundaries. A service-oriented approach not only standardizes this approach, but also allows for greater flexibility in the process. The trend is the desire for companies to easily provide services, share data, and use services from other businesses.

With an SOA solution, your programs can be on different systems and be provided by different vendors, and yet communicate and exchange data with each other. By deploying Web services, valuable CICS applications can evolve to participate in new, more flexible business models.

SOA is an evolution of best practices and technologies, combining the developments made in internet-based technology and inter-operability standards. This results, among other things, a more user-friendly environment for application development and integration of existing business IT assets into a distributed intra and inter-company solution.

This integrated architecture approach is based on the concept of services and allows business logic to be separated from application logic. While it is not a formal specification in itself, imagine a distributed solution that provides application functions across both internal business units and corporations. These functions are delivered as services that may include business logic, application logic, or a combination of both. The result is that flexible systems can be built that implement the changing business processes quickly by making use of reusable components.

The definition of SOA can be further summarized as follows:

- A set of business-aligned IT services that support an organization’s business goal and objectives.
- A set of architectural principles that address characteristics such as modularity, loose coupling, and separation of functions.
- An architectural style that requires a service provider, a service consumer, and a service description.
- A set of services that can be combined and choreographed.
- A programming model that comes with standards, tools, methods, and technologies, such as Web Services.

2.2 Basic components of an SOA solution

At the most basic level, an SOA solution consists of the following three components and is shown in Figure 2-1.

- **Service provider** - creates a service and publishes its interface and access information to the service registry.
- **Service requester** - a client invocation that requires the service by binding to the service provider and invoking an action on the service exposed by the provider.
- **Service registry** - also known as a service broker, makes the service interface and implementation access the information that is available to the service requesters.

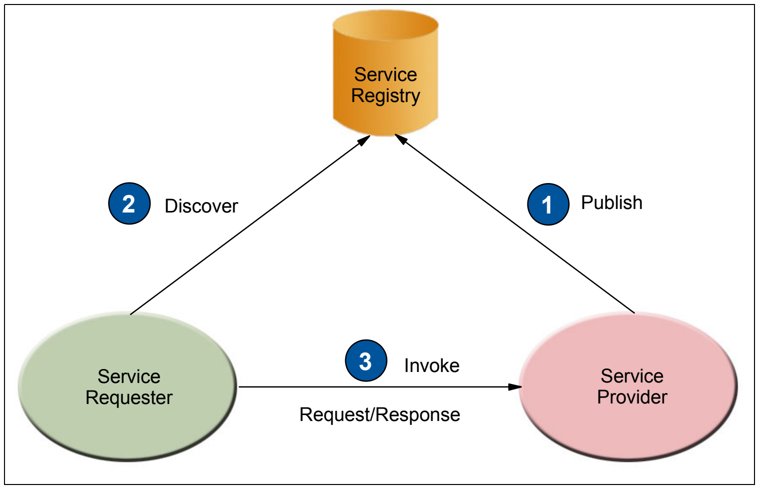


Figure 2-1 SOA components and operations

Now we can define a service as being a function that can be offered or provided to a requester. The scope of this can be a single business function or part of a collection of business functions that are wired together to form a process.

Other, crucial aspects to a service that must be considered when developing an SOA solution are the following commonly agreed-on aspects:

- Services should encapsulate a reusable business function
- Services are defined by explicit, implementation-independent interfaces
- Services are invoked through communication protocols that use location transparency and interoperability

An example of a reusable service that is accessible by more that one requesting application is one that offers a calculation such as a quote for a car part stock order. This could be requested by many different programs or clients from both within the enterprise and from third parties. The reusability of the service in this case will be correct as long as the interfaces of the component that offers the service are clearly defined.

For more information about the relationship between Web services and service -oriented architectures, refer to the IBM Redbooks publication *Patterns: Service-Oriented Architecture and Web Services*, SG24-6303.

2.3 Web services

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network.

Web services are becoming the standard for basic SOA implementation by taking advantage of existing open-standard Web technologies such as XML, URL, and HTTP. These comprise a set of standards that facilitate open system-to-system communication. By adhering to Web services, applications that are based on different platforms and technologies can cooperate through well defined interfaces. Web services follow the SOA philosophy of loose coupling between service requesters and providers.

The formal definition of a Web service, provided by the World Wide Web consortium (W3C) Services Architecture Working Group is as follows:

"*A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web services in a manner prescribed by it's description using SOAP messages, typically conveyed using HTTP with and XML serialization in conjunction with other Web-related standards.*"

2.3.1 Properties of a Web service

While this is widely documented, it can be summarized that all Web services share the following properties:

- **Web services are self contained**. On the client side, no additional software is required. A programming language with XML and HTTP client support is a minimum requirement. On the server side, an HTTP server and a SOAP server are required.

- **Web services are self-describing.** A Web Service Description Language (WSDL) file provides all the information needed to implement a Web service as a provider, or to invoke a Web service as a requester.
- **Web services can be published, located, and invoked across the Web**. Existing infrastructure by way of established lightweight internet standards such as HTTP are used.
- **Web services are modular**. Simple Web services can combined to form more complex implementations, shortening development time.
- **Web services are language-independent and interoperable**. The client and server can be implemented in different environments and operating system platforms. Any language can be used to implement Web service clients and servers.
- **Web services are inherently open and standards-based**. XML and HTTP are the major technical foundation. A large part of the Web service technology was built using open source methodology, maintaining vendor independence.
- **Web services are loosely coupled**. A service requester has to know the interface to a Web service, but not the details of how it was implemented.
- **Web services provide the ability to wrap existing applications**. This is done by providing a Web service as an interface to the application.

2.3.2 Web service standards

Web services standards and specifications change rapidly to keep up with emerging technologies. Figure 2-2 on page 16 provides a summary of the standards today.

Then, within each standard there are a number of specifications that we will outline. Section 2.3.3, “WS standards in CICS TS” on page 18 discusses the specifications of interest to our book and introduces terminology relevant to transforming or developing CICS assets to incorporate this technology.

Given the pace at which the standards are evolving, there is an online compilation of Web services standards at the following Web site:

`http://www-128.ibm.com/developerworks/webservices/standards`

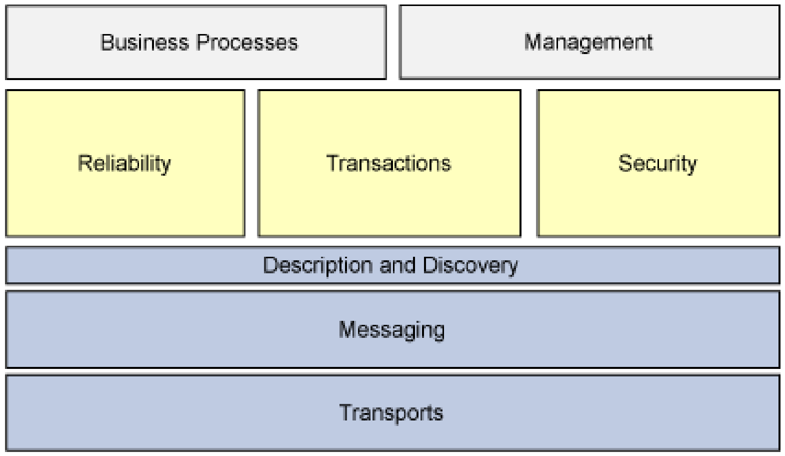


Figure 2-2 Core standards of Web services

Figure 2-2 outlines the eight standards of Web service, which we discuss further in the following list.

1. **Business processes**

   A business process specifies the potential execution order of operations from a collection of Web services, the data shared between these Web services, which partners are involved, and how they are involved in the business process. It also includes joint exception handling for collections of Web services and other issues involving how multiple services and organizations participate. BPEL (Business Process Execution Language) specifies business processes and how they relate to Web services.

2. **Management**

   Web services manageability is defined as a set of capabilities for discovering the existence, availability, health, performance, usage, as well as the control and configuration of a Web service within the Web services architecture. As Web services become pervasive and critical to business operations, the task of managing and implementing them is imperative to the success of business operations.

3. **Reliability**

   It is not possible to solve business issues if the participants are unable to be sure of the completion of message exchanges. Reliable messaging, which allows messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures, is therefore critical to Web services.

4. **Transactions**

   Transactions are a fundamental concept in building reliable distributed applications. A Web service environment requires coordination behavior provided by a traditional transaction mechanism to control the operations and outcome of an application.

   The specifications include the following:

   - Web services atomic transactions
   - Web services business activity
   - Web services co-ordination

5. **Security**

   There are a number of specifications that applications can engage in secure communication that are designed to work with the general Web services framework. Following are some of these specifications:

   - Web services Federation Languages
   - Web services Trust
   - Web services Security policy
   - Web services Secure Conversation Language

6. **Description and discovery**

   Web services are meaningful only if potential users may find information sufficient to permit their execution. The focus of these specifications and standards is the definition of a set of services supporting the description and discovery of businesses, organizations, and other Web services providers, the Web services they make available, and the technical interfaces which may be used to access those services.

   Amongst others, some of these specifications are discussed in a following section and pertain to the CICS implementation of Web services:

   - UDDI
   - WSDL

7. **Messaging**

   These messaging standards and specifications are intended to give a framework for exchanging information in a decentralized, distributed environment. CICS TS provides support for the SOAP specification, which we explore in later chapters.

8. **Transports**

   BEEP, the Blocks Extensible Exchange Protocol (formerly referred to as BXXP), is a framework for building application protocols. It was standardized by IETF and does for Internet protocols what XML does for data.

2.3.3 WS standards in CICS TS

Certainly CICS TS adheres to many of the specifications outlined in the previous section, and the following are of particular interest to the development of Web services in CICS:

- Description and Discovery - in particular UDDI and WSDL.
- WS-Transactions - the family of specifications that relate to transactional Web services.
- WS-Security - the family of specifications that relate to securing Web services.
- Messaging - the SOAP for CICS feature and incorporation of SOAP message handlers.

Following is additional terminology that describes some key specifications that are useful within the scope of this publication:

XML

Extensible Markup Language or XML is the foundation of Web services. However, since much information is already written about XML, we do not describe it here. The following document provides a good reference about this standard:

`http://www.w3.org/XML/`

SOAP

SOAP provides an XML, text-based platform and language neutral message format. Originally proposed by Microsoft®, SOAP was designed to be a simple and extensible specification for the exchange of structured XML-based information in a decentralized, distributed environment. As such, it represents the main means of communication between the three functional components in an SOA: the service provider, the service requester, and the service registry.

There are currently two versions of SOAP: Version 1.1 and Version 1.2.

The SOAP specification consists of the following three parts:

i. An ***envelope*** that defines a framework for describing message content and processing instructions. Each SOAP message consists of an envelope that contains a number of headers and one body that carries the payload, or data, to exchange. SOAP messages may also contain faults that report failures or unexpected conditions.

ii. A set of ***encoding rules*** for expressing instances of application-defined data types.

iii. A ***convention*** for representing remote procedure calls and responses.

A SOAP message is, in principle, independent of the transport protocol that is used: HTTP, JMS, SMTP, or FTP. The most common way of exchanging SOAP messages is through HTTP.

WSDL

Web services Description Language (WSDL) uses XML to specify the characteristics of a Web service, what the Web service can do, where it resides, and how it is invoked. WSDL can be extended to allow descriptions of different bindings, regardless of what message formats or network protocols are used to communicate.

WSDL enables a service provider to specify the following characteristics of a Web service:

- The **name** of the Web service and addressing information.
- The **protocol and encoding style** to be used when accessing the public operations of the Web service.
- The **Type information**: operations, parameters, and data types comprising the interface of the Web service, including a name for this interface.

WSDL is not bound to any protocol or network service. It can be extended to support many different message formats and network protocols. However, because Web services are mainly implemented using SOAP and HTTP, the corresponding bindings are part of this standard.

UDDI

The Universal Description, Discovery, and Integration standard defines a means to publish and to discover Web services. The current release is UDDI Version 3.0. For more information, refer to the following Web sites:

http://www.uddi.org/
http://www.oasis-open.org/specs/index.php#wsssv1.0

Further details on these standards reside in the IBM Redbooks publication, *Implementing CICS Web Services*, SG24-7206.

2.3.4 Implementing Web services

Figure 2-3 shows the model implemented by Web services.

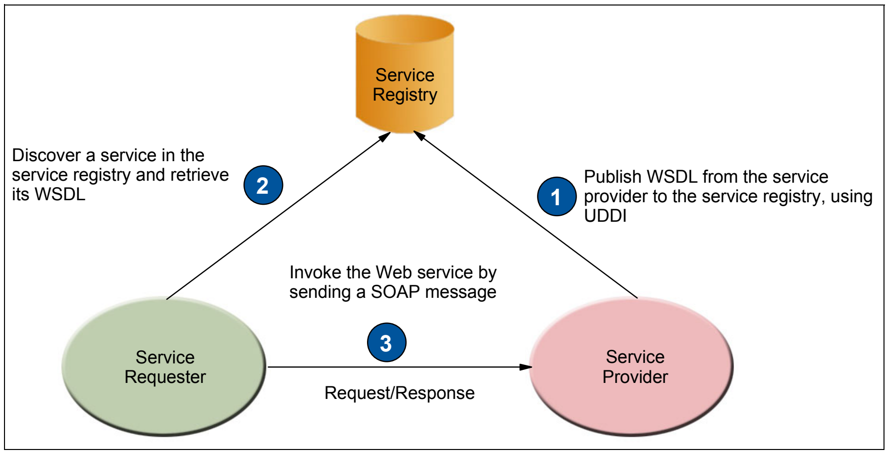

Service Registry
Discover a service in the service registry and retrieve its WSDL
2
1
Publish WSDL from the service provider to the service registry, using UDDI
Invoke the Web service by sending a SOAP message
3
Service Requester
Service Provider
Request/Response

Figure 2-3 Web services invocation model

Following is a description of the interaction in Figure 2-3:

1. The service provider publishes the Web Services Description Language (WSDL) data that defines its interface and location to a service registry such as the Universal Description, Discovery, and Integration (UDDI) service registry.
2. The service requester contacts the service registry to obtain reference to a service provider.
3. The service requester, having obtained the location of the service provider, makes calls on the provider by sending a SOAP-formatted message.

Web service Usage Models

Basic Web services support provides the following three simple usage models:

- **One-way usage scenario** - a Web services message is sent from a requester to a provider, and no response message is expected.
- **Synchronous request or response usage scenario** - a message is sent from a requester to a provider and a response message is expected.
- **Basic callback usage scenario** - a message is sent from a requester to a provider using the 2-way invocation model, but the response is treated only as an acknowledgement of a request having been received. The provider then responds by using a Web service callback to the requester.

Other Web service standards are built upon these basic standards and invocation models to provide higher level functions and qualities of service.

We will show later on how our sample application fits within this framework.

2.4 Implementing SOA on z/OS

SOA provides an increasing number of options for accessing existing assets on z/OS. In a sense, the mainframe environment has always lent itself to the concept of SOA because it regards all it's resources as providing services. Resources specifically designed for SOA capabilities are such components as the Enterprise Service Bus (ESB), process management engines (WPS), base Java™ 2 Enterprise Edition (J2EE™), WebSphere Application Server, and databases.

To offer the power of System z™ for SOA, IBM has developed specific z/OS versions of its SOA product suite that is built on IBM WebSphere Application Server V6 for z/OS. The WebSphere Process Server and WebSphere Enterprise Service Bus are z/OS-enabled, as are supporting components such as IBM DB2® for z/OS V8. This offers a clean and contained architecture within a z/OS environment with the architecture based on open and interoperability standards.

CICS Transaction Server for z/OS V3.1 has added features to support SOA technologies such as Web Services and can integrate with the WebSphere Application Server for z/OS-based products. We will demonstrate in Chapter 6 of this book how CICS can be used as a service provider within the SOA framework and the interaction with the various products and components within a larger, distributed environment.

2.5 Realizing that CICS assets can be SOA solutions

Part of the hesitation in using new technology to transform existing assets in the form of heritage applications into an SOA solution is the seemingly complex relationship between all the components involved or even in knowing what components need to be invoked.

The scenario we are using in this publication is by no means the only solution; instead, it provides one way of using some of the technology to achieve a robust SOA implementation. More of this is discussed in the chapters that follow, but there is first a need for the application programmer to recognize the types of CICS assets that can be transformed.

CICS programs are typically grouped into application suites or components for performing a common set of business actions. Identifying the CICS programs that provide flexible public interfaces and understanding these interfaces is the first key step for re-use.

Over the past 36 years, developers have created three major types of CICS applications or assets:

1. **CICS COMMAREA programs**

   These programs receive requests and send responses through an area of storage called the COMMunications AREA (COMMAREA). The programs themselves can be written in COBOL, PL/I, C, C++, Assembler, or Java. These programs are akin to subroutines as they are unaware of how they were invoked. Their state, transactional scope, and security context are managed by CICS itself.

2. **CICS terminal-oriented programs**

   These programs are sometimes referred to as 3270 programs because they were designed to be invoked directly from an IBM 3270 Display Station or similar buffered terminal device. Invocation usually corresponds to a single interaction in a user dialog, starting with the receipt of a message from the terminal and ending with the transmission of a reply message to the same device.

   Input data from the terminal device is carried in a datastream, which the application acquires through a RECEIVE command. After processing, an output datastream is transmitted back to the terminal device through a SEND command.

   Terminal-oriented programs must be capable of analyzing device-specific input data streams and building the output data streams that are to be transmitted to the terminal.

   With the introduction of *basic mapping support* (BMS), the programmer only needed to be concerned with the static layout of the panel to be displayed. Device-specific information and terminal datastreaming was handled by BMS, thus enabling applications to be more device independent.

3. **CICS programs that use channels and containers**

   Channels and containers are new resources in CICS TS V3.1 that provide the capability to pass data from one application to another application.

- A *channel* is a logical resource that must contain one or more containers.
- A *container* is a named block of data designed for passing information between programs.

The major advantage of using channels and containers compared to using a COMMAREA is that the length of a container can exceed the 32 KB limit for COMMAREA data. CICS uses channels and containers to pass data between the message handlers of a pipeline. This is discussed further in section 2.7, “Channels and containers” on page 25.

2.6 Access to COMMAREA programs

The best practice in CICS application design for a number of years now has been to separate key elements of the application into the following sub-categories:

- Client adapt or presentation logic
- Integration logic
- Business logic
- Data access logic

Figure 2-4 shows a transaction made up of these separate components. A COMMAREA or ‘channel’ interface is used to pass data between the components.

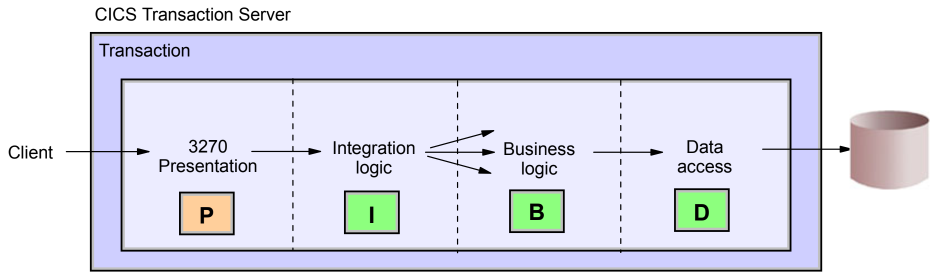


Figure 2-4 The key application elements

This separation provides a framework that enables the following:

- Reuse of business logic and data access logic programs as sub-routines within a larger application
- Reuse with alternative implementations of presentation logic—for example a Web Service, a Web browser, or a 3270 device

CICS COMMEARA programs can be relatively easily enabled for access from a variety of different client applications running on a wide variety of platforms—distributed servers and mainframes. Typical clients include the following:

- Web service requester
- Java servlet or Enterprise JavaBeans™ (EJB™) running on a J2EE application server
- An application running on a Microsoft .NET environment
- Web browser
- Messaging Middleware - WebSphere MQ

In most cases, connections from a client will use a combination of the following:

- Internal adapters
- External connectors
- Standard IP-based protocols

An **internal adapter** is simply a program that accepts a request and converts the data from an external format to the internal format used by the CICS business logic. CICS Web Support introduced this concept with the converter program DFHWBTTA. Using a more recent technology, an adapter can convert a SOAP message to a COMMAREA format. The transport mechanism used to invoke the adapter may be synchronous or asynchronous.

An **external connector** provides a remote call interface and implements a private protocol to invoke an application running under CICS TS. An external adapter converts data from its external form to the COMMAREA format. CICS Transaction Gateway is the most well known example of an external connector that implements the Common Connector Interface specified by the J2EE Connector Architecture (JCA).

The **standard IP-based adapters** that use a specific transport are IBM WebSphere MQ, HTTP, and TCP/IP sockets. These methods permit greater flexibility in the functionality that can be implemented. We decided to use IBM WebSphere MQ to demonstrate access to traditional CICS programs from an external client.

2.6.1 Access to terminal-oriented programs

There are many programs that do not have such a clear separation of logic as COMMAREA programs, for which there is only a 3270 interface. CICS TS3.1 provides a LINK3270 bridge function that simulates a client actually interfacing directly with a 3270 screen. This is achieved by the client making a connection to the Link3270 bridge in CICS TS (program DFHL3270) and then passing a COMMAREA that includes a transaction identifier and the data that needs to be passed to the application. The response then contains the 3270 screen data reply, and the information is presented back to the client.

We do not discuss the development and implementation of terminal-oriented programs any further in this book.

2.7 Channels and containers

Essentially the use of channels and containers provides a solution to the 32KB limit imposed on the traditional CICS COMMAREA in order to accommodate modern applications. There is now a need for considering how you currently handle data exchange and whether implementing this new function will benefit your application design needs.

Consider some of the COMMAREA issues you may face when handling large data objects:

- Applications must use a circumvention technique, such as using external VSAM files or splitting the data into separate parts. This method increases risk as well as programming time and effort.
- Passing XML documents by value throughout the request process path becomes inhibited because the size constraint applies to the following:
  - Calls between CICS programs both within the local system and between CICS systems
  - Parameter data passed between CICS tasks
  - External client programming interfaces such as CICS interface (EXCI and the CICS client external call interface (ECI)
- Data structures used to define a COMMAREA payload can become overloaded. Redefining structures on the same area of memory increases the risk of program errors. Similarly, confusion about the validity of fields can result in application programming errors.
- An overloaded COMMAREA structure increases transmission time between CICS regions because the structure size must account for the maximum size of the data that could be returned from the called program—and this parameter size depends on the request logic invoked.

  CICS TS must always allocate memory to accommodate the return of the maximum COMMAREA structure size.
- A code-page conversion of COMMAREA structure is complex because binary and character data cannot be easily separated.

2.7.1 Advantages over COMMAREAs

The containers and channels approach has several advantages over COMMAREAs:

- Containers can be any size and, as a result, can extend beyond the maximum 32KB size of a COMMAREA. There is no limit to the number of containers that can be added to a channel, and the size of the individual containers is limited only by the amount of storage available.
- A channel consists of multiple containers, enabling it to be used to pass data in a more structured way. In contrast, a COMMAREA is a single block of data.
- Unlike COMMAREAs, channels do not require the programs that use them to know the exact size of data returned, making programming easier.

2.7.2 Channels

A channel is a uniquely named reference to a collection of application parameter data held in containers. Its analogous to a COMMAREA but is not subject to the constraints of a COMMAREA.

You can choose a channel name that is a meaningful representation of the data structures that the channel is associated with. For example in a human resource application a channel name might be <employee-info>.

This collection of application parameter data serves as a standard mechanism to exchange data between CICS programs.

CICS TS provides an EXEC API that associates a named channel with a collection of one or more containers—offering an easy way to group parameter data structures to pass to a called application.

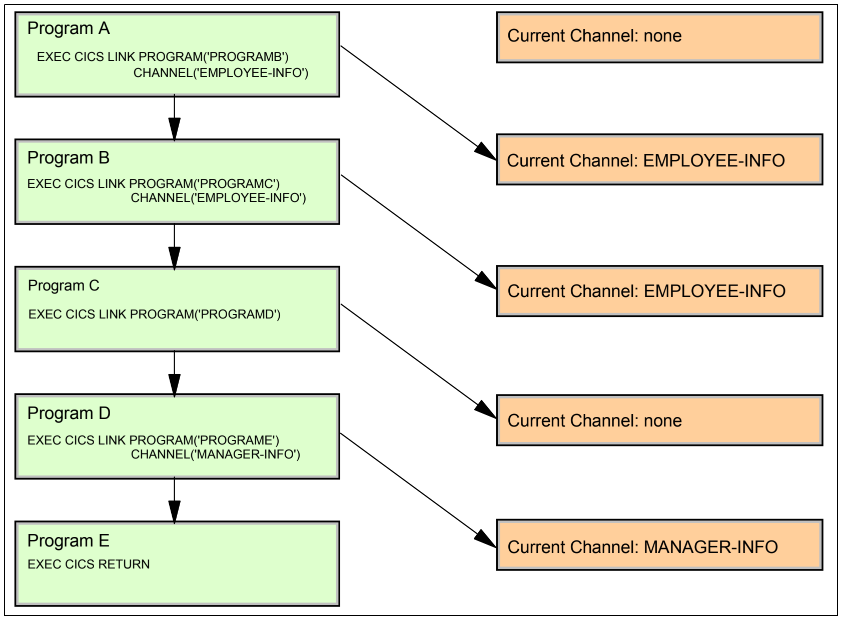
CICS TS removes a channel when it can no longer be referenced—when it becomes out of scope.

The current channel

A program's current channel is the channel (if any) with which it was invoked. The current channel is set by the calling program or transaction by transferring the control to the called program via a LINK, XCTL, START, and pseudo-conversational RETURN with the channel parameter.

Although the program can create other channels, the current channel, for a particular invocation of a particular program, never changes. It is analogous to a parameter list.

If a channel is not explicitly specified, the current channel is used as the default value for the CHANNEL (channel-name) parameter on the EXEC CICS command. This is shown in Figure 2-5.



Program A
EXEC CICS LINK PROGRAM('PROGRAMB')
CHANNEL('EMPLOYEE-INFO')
Program B
EXEC CICS LINK PROGRAM('PROGRAMC')
CHANNEL('EMPLOYEE-INFO')
Program C
EXEC CICS LINK PROGRAM('PROGRAMD')
Program D
EXEC CICS LINK PROGRAM('PROGRAME')
CHANNEL('MANAGER-INFO')
Program E
EXEC CICS RETURN
Current Channel: none
Current Channel: EMPLOYEE-INFO
Current Channel: EMPLOYEE-INFO
Current Channel: none
Current Channel: MANAGER-INFO

Figure 2-5 The current channel

Typically, programs that exchange a channel are written to handle that channel. That is, both client and server programs know the name of the channel and the names and number of the containers in that channel. However, if, for example, a server program or component is written to handle more than one channel, on invocation it must discover which of the possible channels it was passed.

A program can discover its current channel—that is, the channel with which it was invoked—by issuing an **`EXEC CICS ASSIGN CHANNEL`** command. (If there is no current channel, the command returns blanks.)

The program can also retrieve the names of the containers in its current channel by browsing, but typically, this is not necessary. A program written to handle several channels is often coded to be aware of the names and number of the containers in each possible channel.

To get the names of the containers in the current channel, use the **`browse`** commands as shown in Example 2-1 on page 28.

Example 2-1 Browsing containers in a channel

```
EXEC CICS STARTBROWSE CONTAINER BROWSETOKEN(data-area)
EXEC CICS GETNEXT CONTAINER(data-area) BROWSETOKEN(token)
EXEC CICS ENDBROWSE CONTAINER BROWSETOKEN(token)
```

Having retrieved the name of its current channel and, if necessary, the names of the containers in the channel, a server program can adjust its processing to suit the kind of data that it was passed.

Note: For a program creating a channel, the `ASSIGN CHANNEL` command will return blanks unless it was invoked via START, LINK, or XCTL specifying the channel name.

The scope of a channel

The scope of a channel is the code (for example, the program or programs) from which it can be accessed.

Figure 2-6 on page 29 shows the scope of channel EMPLOYEE-INFO, which consists of programs A (the program which created it), program B (for which it is the current channel), and program C (for which it is also the current channel). Additionally, we show the scope of channel MANAGER-INFO, which consists of programs D (which created it) and Program E (for which it is the current channel).

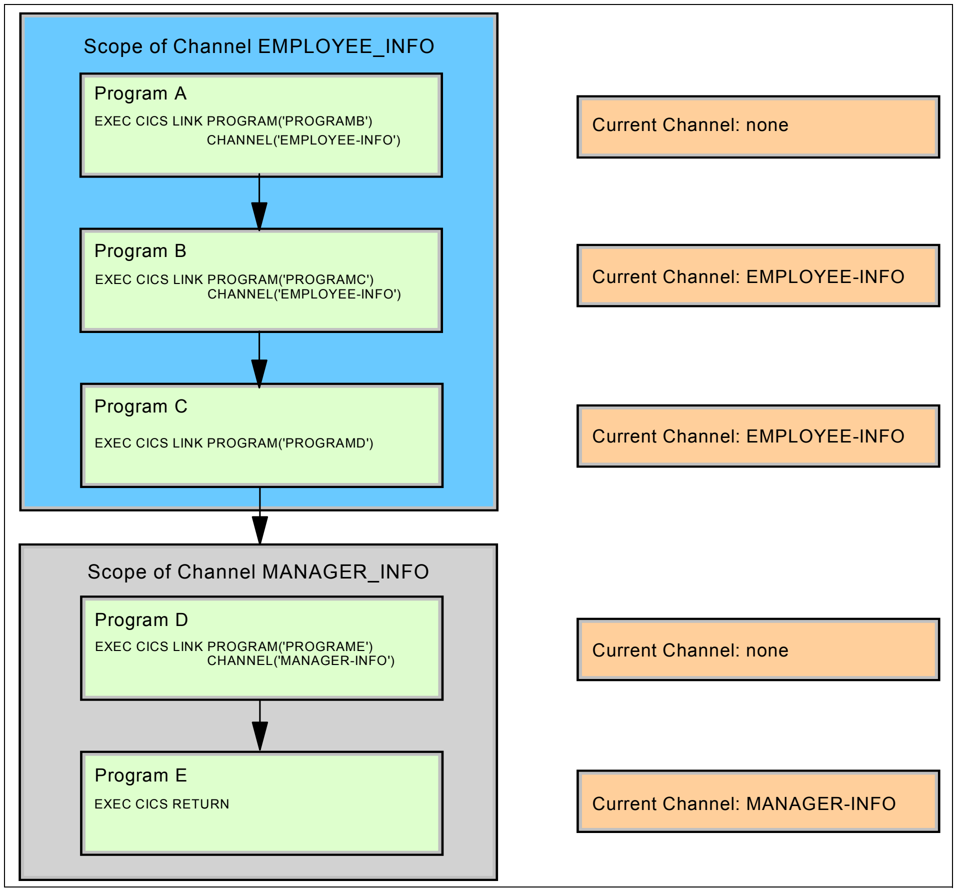

Scope of Channel EMPLOYEE_INFO
Program A
EXEC CICS LINK PROGRAM('PROGRAMB') CHANNEL('EMPLOYEE-INFO')
Program B
EXEC CICS LINK PROGRAM('PROGRAMC') CHANNEL('EMPLOYEE-INFO')
Program C
EXEC CICS LINK PROGRAM('PROGRAMD')
Scope of Channel MANAGER_INFO
Program D
EXEC CICS LINK PROGRAM('PROGRAME') CHANNEL('MANAGER-INFO')
Program E
EXEC CICS RETURN
Current Channel: none
Current Channel: EMPLOYEE-INFO
Current Channel: EMPLOYEE-INFO
Current Channel: none
Current Channel: MANAGER-INFO

Figure 2-6 Example showing the scope of a channel

Lifetime of a channel

A channel is created when it is named on an EXEC CICS command. The usual command to create a channel is the **`EXEC CICS PUT CONTAINER`** command, in which specifying the CHANNEL parameter creates the channel and also associates the container with it.

A channel is deleted when it goes out of scope to the programs in the linkage stack, meaning that no programs can access it. This will cause the channel to be deleted by CICS.

Figure 2-7 on page 30 shows the APIs used to create and manage a channel.

- **EXEC CICS PUT CONTAINER CHANNEL**
  Creates a channel and places data into a container within the channel
- **EXEC CICS GET CONTAINER CHANNEL**
  Retrieves the container data passed to the called program
- **EXEC CICS MOVE CONTAINER CHANNEL AS TOCHANNEL**
  Moves a container from one channel to another channel
- **EXEC CICS DELETE CONTAINER CHANNEL**
  Deletes a container
- **EXEC CICS ASSIGN CHANNEL**
  Returns the name of the program's current channel, if one exists
- **EXEC CICS LINK PROGRAM CHANNEL**
  Links to the program, on a local or remote system, passing the channel and container data
- **EXEC CICS XCTL PROGRAM CHANNEL**
  Transfers control to the program passing the channel and container data
- **EXEC CICS START TRANSID CHANNEL**
  Starts a task, on a local or remote system, copying the named channel and container data and passing it to the started task
- **EXEC CICS RETURN TRANSID CHANNEL**
  Returns control to CICS, passing the channel and container data to the next transaction

Figure 2-7 API to create and manage a channel

2.7.3 Containers

A container is a uniquely named block of data that can be passed to a subsequent program or transaction. It refers to a particular parameter data structure that exists within a collection of virtually any form of application parameter data.

You can choose a container name that is a meaningful representation of the data structure. For example, in a human resource application, the container name might be <employee-name>.

CICS TS provides EXEC API verbs to create, delete, reference, access, and manipulate a container as well as to associate it with a channel. See Figure 2-8 on page 31 below for more details.

```
► EXEC CICS PUT CONTAINER CHANNEL
Creates a channel and places data into a container within the channel
► EXEC CICS GET CONTAINER CHANNEL
Retrieves the container data passed to the called program
► EXEC CICS MOVE CONTAINER CHANNEL AS TOCHANNEL
Moves a container from one channel to another channel
► EXEC CICS DELETE CONTAINER CHANNEL
Deletes a container from a channel
► EXEC CICS STARTBROWSE CONTAINER
Start a browse of the containers associated with a channel
► EXEC CICS GETNEXT CONTAINER
Return the name of the next container associated to the channel
► EXEC CICS ENDBROWSE CONTAINER
Ends the browse of the containers associated with the channel
```

Figure 2-8 Container related API

A container can be any length, and a container size is constrained only by the available user storage in the CICS address space.

It can include data in any format required by an application. An application can create any number of containers and can use separate containers for different data types, such as binary and character data. This capability helps ensure that each container structure is based on a unique area of memory.

It also minimizes the potential for errors that commonly arise when parameter data for multiple applications is overloaded in a single memory area, by isolating different data structures, and making the association between data structure and purpose clear.

CICS read-only containers

CICS can create channels and containers for its own use and pass them to user programs. In some cases CICS marks these containers as read-only, so that the user program cannot modify data that CICS needs to return from the user program.

User programs cannot create read-only containers.

You cannot overwrite, move, or delete a read-only container. Thus, if you specify a read-only container on a **PUT CONTAINER**, **MOVE CONTAINER**, or **DELETE CONTAINER** command you will receive an INVREQ condition.

Note: Channel containers are not recoverable. If you need to use recoverable containers, use CICS business transaction services (BTS) containers.

2.7.4 Data conversion

The data conversion model used by channel applications is much simpler than the data conversion model used by COMMAREA applications. This is because data conversion in COMMAREA applications is controlled by the system programmer, whereas in channel applications it is controlled by the application programmer using simple API commands.

Here are some cases where data conversion is necessary:

- When character data is passed between platforms that use different encoding standards—for example, EBCDIC, and ASCII.
- When you want to change the encoding of some character data from one Coded Character Set Identifier (CCSID) to another.

Applications that use Channels to exchange data use a simple data conversion model. Frequently, no conversion is required, but when conversion is required, a single programming instruction can be used to tell CICS to handle it automatically.

Using COMMAREAs

For applications that use the COMMAREAs to exchange data, the conversion is done under the control of the system programmer using the DFHCNV conversion table, the DFHCCNV conversion program, and optionally the DFHUCNV user-replaceable conversion program.

Using channels

The data conversion model used by channel applications is much simpler than that used by the COMMAREA applications. The data in channels and containers is converted under the control of the application programmer using API commands.

- The application programmer is responsible only for the conversion of user data—that is, the data in containers created by the application programs. System data is converted automatically by CICS, where necessary.
- The application programmer is concerned only with the conversion of character data. The conversion of binary data (between big-endian and little-endian) is not supported.
- Applications can use the container API as a simple means of converting character data from one code page to another. Example 2-2 on page 33 converts data from codepage1 to codepage2:

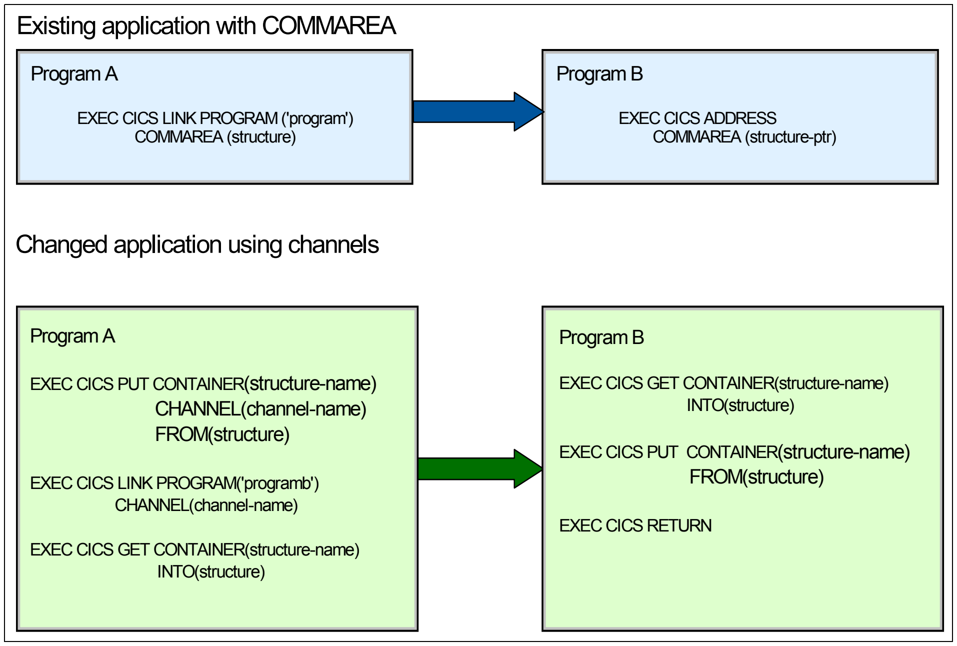
Example 2-2 API to convert codepage

```
EXEC CICS PUT CONTAINER(temp) DATATYPE(CHAR)
              FROMCCSID(codepage1) FROM(input-data)
EXEC CICS GET CONTAINER(temp) INTOCCSID(codepage2)
              SET(data-ptr) FLENGTH(data-len)
```

2.7.5 Migrating COMMAREA to channels and containers

To migrate programs exchanging data via a COMMAREA on a LINK command, the format of the command must be changed and proper commands must be added to use channels and containers.

Figure 2-9 shows an example of this.


Existing application with COMMAREA
Program A
Program B
Changed application using channels
Program A
Program B

```
Existing application with COMMAREA

Program A
    EXEC CICS LINK PROGRAM ('program')
         COMMAREA (structure)

Program B
    EXEC CICS ADDRESS
         COMMAREA (structure-ptr)

Changed application using channels

Program A
EXEC CICS PUT CONTAINER(structure-name)
              CHANNEL(channel-name)
              FROM(structure)

EXEC CICS LINK PROGRAM('programb')
         CHANNEL(channel-name)

EXEC CICS GET CONTAINER(structure-name)
              INTO(structure)

Program B
EXEC CICS GET CONTAINER(structure-name)
              INTO(structure)

EXEC CICS PUT  CONTAINER(structure-name)
               FROM(structure)

EXEC CICS RETURN
```

Figure 2-9 Changes from commarea to channels using LINK

The same applies to programs using the START command with the COMMAREA. Figure 2-10 on page 34 shows an example of this.

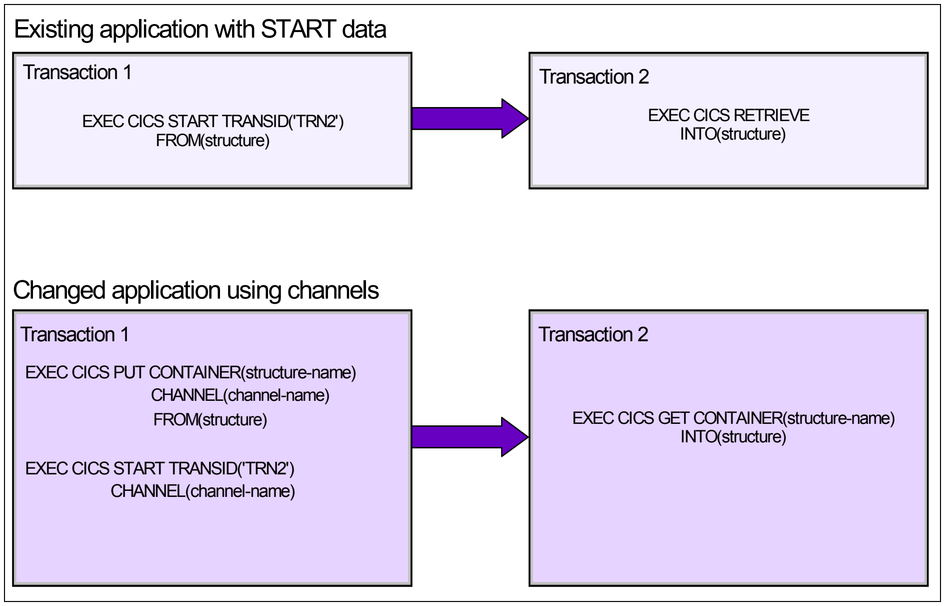

Existing application with START data
Transaction 1
EXEC CICS START TRANSID('TRN2') FROM(structure)
Transaction 2
EXEC CICS RETRIEVE INTO(structure)
Changed application using channels
Transaction 1
EXEC CICS PUT CONTAINER(structure-name) CHANNEL(channel-name) FROM(structure)
EXEC CICS START TRANSID('TRN2') CHANNEL(channel-name)
Transaction 2
EXEC CICS GET CONTAINER(structure-name) INTO(structure)

Figure 2-10 Changes from commarea to channels using START

Migration consideration

Following is a list of items you may want to consider when migrating from a COMMAREA to channels and containers:

- CICS application programs that use traditional COMMAREAS to exchange data will continue to work as before.
- **`EXEC CICS LINK`** and **`EXEC CICS START`** commands, which can pass either COMMAREAs or channels, can be dynamically routed.
- If you employ a user-written dynamic or distributed routing program for workload management, rather than CICSPlex SM, you must modify your program to handle the new values that it may be passed in the DYRLEVEL, DYRTYPE, and DYRVER fields of the DFHDYPDS communications area.
- It is possible to replace a COMMAREA by a channel with a single container. While this may seem the simplest way to move from COMMAREAs to channels and containers, it is not good practice to do this.
- Also, be aware that a channel may use more storage than a COMMAREA designed to pass the same data. Because you are taking the time to change your application programs to exploit this new function, you should implement the "best practices" for channels and containers.
- Channels have several advantages over COMMAREAs, and it pays to design your channels to make the most of these improvements.

- In previous releases, because the size of COMMAREAs is limited to 32K and channels were not available, some applications used temporary storage queues (TSQs) to pass more than 32K of data from one program to another. Typically, this involved multiple writes to and reads from a TSQ. If you migrate one of these applications to use channels, be aware of the following:
  - If the TSQ used by your existing application is in main storage, the storage requirements of the new, migrated application are likely to be similar to those of the existing application.
  - If the TSQ used by your existing application is in auxiliary storage, the storage requirements of the migrated application are likely to be greater than those of the existing application. This is because container data is held in storage rather than being written to disk.

Additional information can be found in the Redbooks publication *CICS Transaction Server V3R1 Channels and Containers Revealed*, SG24-7227.

2.8 Web services support in CICS TS V3.1

In CICS TS 3.1, support was added for Web services, meaning that applications running in a traditional CICS environment could now participate in a Web services environment as either service providers, requesters, or both. Years of investment into application design and implementation are now increasing their value by being able to participate in new technologies without the need for re-programming.

This section discusses the support added to CICS to enable a Web service environment. It is intended as a summary, as there are already IBM Redbooks publications written entirely on this topic. See the following Redbooks publications:

- *Implementing CICS Web Services*, SG24-72061
- *Securing Access to CICS Within an SOA*, SG24-57561
- *Application Development for CICS Web Services*, SG24-7126

Some of the Web Services functions in CICS TS 3.1 are now summarized.

2.8.1 Web services assistant utility

This utility contains two programs, DFHWS2LS and DFHLS2WS:

- DFHWS2LS helps you to map an existing WSDL document into a high level programming language data structure.

- DFHLS2WS helps you to create a new WSDL document from an existing language structure.

The Web services assistant supports the COBOL, PL/I, C, and C++ programming languages.

2.8.2 Deploying CICS applications

The Web services support allows you to take two different approaches for deploying CICS applications:

- **Using the Web services assistant**. This approach helps you to deploy an application with the least amount of programming effort. For example, if you want to expose an existing application as a Web service, you can start with a high-level language data structure, and use DFHLS2WS to generate the Web services description. Alternatively, if you want to communicate with an existing Web service, you can start with its Web service description and use DFHWS2LS to generate the high-level language structure to use in your program.

  Both DFHLS2WS and DFHWS2LS generate a file called the *wsbind file.* When the application runs, CICS will use the wsbind file to transform the application data into a SOAP message on output, and it transforms the SOAP message to application data on input.
- **Write your own code.** To have more control over the processing of your data, you can write your own code to map between your application data and the message that flows between the service provider and service requester. For example, if you want to use non-SOAP messages in the Web service infrastructure, you can write your own code to transform between the message format and the format used by your application.

2.8.3 PIPELINE for message handling

A new concept in CICS TS 3.1 is the pipeline. A *message handler* is a program in which you can perform your own processing of Web service requests and responses. A *pipeline* is a set of message handlers that are executed in sequence.

A pipeline configuration file needs to be created by the CICS systems programmer to determine which message handlers should be invoked in a particular pipeline. It is an XML file that describes both the message handler programs and the SOAP header processing programs that CICS invokes when it processes the pipeline. The pipeline can be configured as either a service requester pipeline or service provider pipeline.

The PIPELINE resource definition is also required, which is used by CICS to handle the Web service request. It contains the name of the pipeline configuration and the location of the WSDL and wsbind files.

2.8.4 Message handlers for SOAP

CICS provides SOAP message handler programs to assist in the configuration of your pipeline as a SOAP node.

- A service requester pipeline is the initial SOAP sender for the request and the ultimate SOAP receiver for the response.
- A service provider pipeline is the ultimate SOA receiver for the request and the initial SOAP sender for the response.

The CICS-provided SOAP message handlers can be configured to invoke one or more user-written SOAP header processing programs and to enforce the presence of particular headers in the SOAP message.

2.8.5 Web services resource definitions

We already discussed the PIPELINE definition, so the following resource definitions are all that is needed to configure support for Web services:

- PIPELINE
- URIMAP
- WEBSERVICE

The application programming interface for these definitions follow the traditional invocations:

- SOAPFAULT ADD | CREATE | DELETE
- INQUIRE WEBSERVICE
- INVOKE WEBSERVICE

3

CICS as a service provider and requester

Having introduced the concepts of SOA and the role that CICS can play in taking advantage in this architecture through support of the Web services technology, we are now going to overview the processing that CICS performs to handle incoming and outgoing Web service requests. We also explain some of the resources that CICS uses to implement the requests.

In a later chapter, we describe how we exposed our sample application as a Web service with CICS as a service provider within this framework.

3.1 Overview of CICS as a service provider

When CICS is a service provider, essentially CICS resources are made available or “exposed” to a request from a client connection from within or external to the enterprise. The request is passed through a CICS pipeline resource to a target application program. The response from the application is then passed back through the same pipeline.

An existing COMMAREA-based application can be exposed as a service provider without any application changes.

Figure 3-1 summarizes the role of CICS as a service provider where the following operations are performed:

1. Receive the request from the service requester.
2. Examine the request and extract the contents that are relevant to the target application program through a pipeline.
3. Invoke the application program, passing data extracted from the request.
4. Construct a response (when the application program returns control) using data returned by the application program.
5. Send a response to the service requester through the same pipeline.

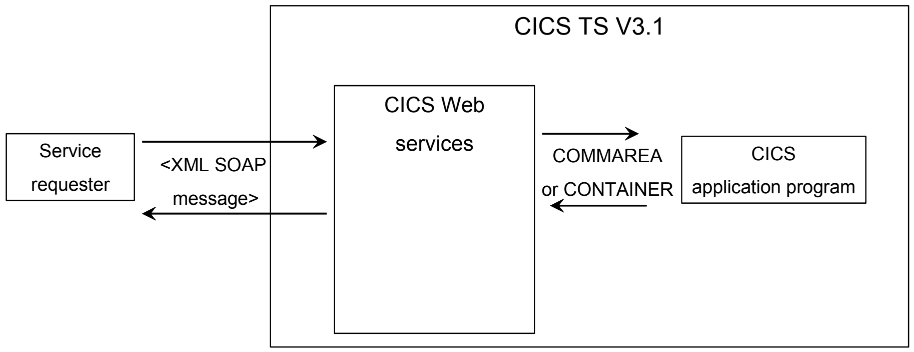


Figure 3-1 CICS as a service provider

3.2 Inbound request processing

Figure 3-2 shows the processing that occurs when a service requester sends a SOAP message over HTTP to a service provider application running in a CICS region.

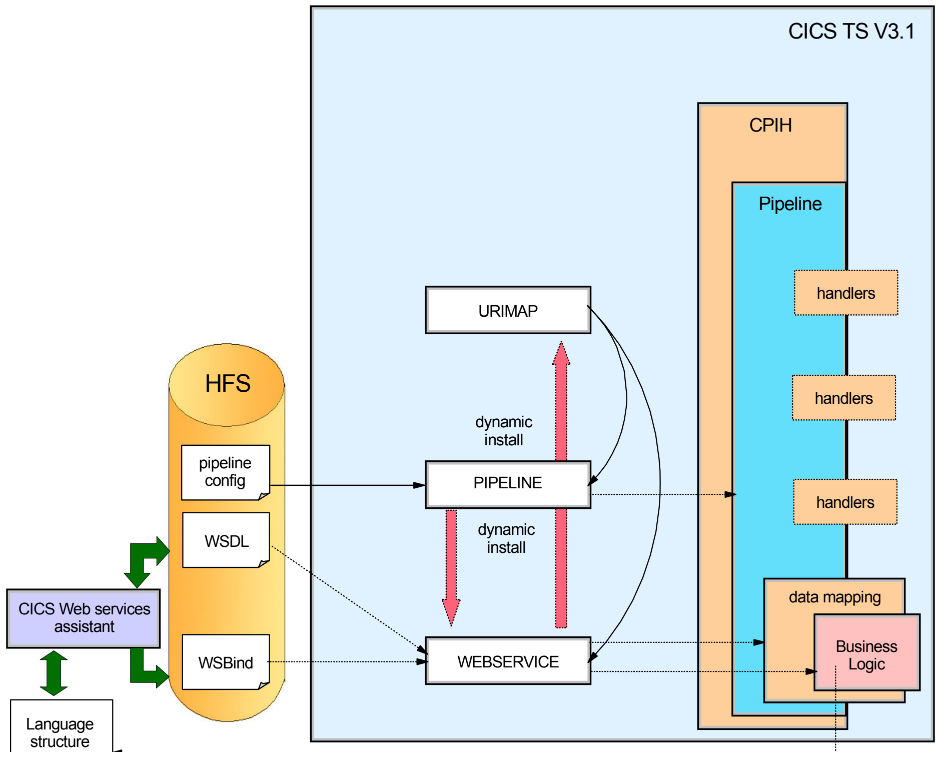


Figure 3-2 Web service run-time service provider processing

The CICS-supplied sockets listener transaction (CSOL) monitors the port specified in the TCPIPSERVICE resource definition for incoming HTTP requests. When the SOAP message arrives, CSOL attaches the transaction specified in the **TRANSACTION** attribute of the TCPIPSERVICE definition. This will, by default be the CICS-supplied Web attach transaction CWXN.

CWXN finds the URI in the HTTP request and then scans the URIMAP resource definitions for a URIMAP that has its **USAGE** attribute set to *PIPELINE* and its **PATH** attribute set to the URI found in the HTTP request. If CWXN finds such a URIMAP, it uses the **PIPELINE** and **WEBSERVICE** attributes of the URIMAP definition to get the name of the PIPELINE and WEBSERVICE definitions, which it uses to process the incoming request.

CWXN also uses the `TRANSACTION` attribute of the URIMAP definition to determine the name of the transaction that it should attach, to process the pipeline. By default this is the CPIH transaction.

CPIH starts the pipeline processing. It uses the PIPELINE definition to find the name of the pipeline configuration file and then determines which message handler programs and SOAP header processing programs to invoke.

A message handler in the pipeline (typically a CICS-supplied SOAP message handler) removes the SOAP envelope from the inbound request and passes the SOAP body to the data mapper function.

CICS uses the DFHWS-WEBSERVICE container to pass the name of the required WEBSERVICE definition to the data mapper. The data mapper uses the WEBSERVICE definition to locate the main storage control blocks that it needs to map the inbound service request (XML) to a COMMAREA or a container.

The data mapper links to the target service provider application program, providing it with input in the form that it expects. The application program is not aware that it is being executed as a Web service. The program performs its normal processing and then returns an output COMMAREA or container to the data mapper.

The output data from the CICS application program cannot just be sent back to the pipeline code. The data mapper must first convert the output from the COMMAREA or container format into a SOAP body.

3.3 Overview of CICS as a service requester

When CICS is a service requester, an application program sends a request, which is passed through a pipeline to a target service provider. The response from the service provider is returned to the application program through the same pipeline. In this section we discuss how to prepare for running a CICS application as a service requester. Then we discuss how CICS processes the outbound service request.

Figure 3-3 on page 43 shows CICS as a service requester.

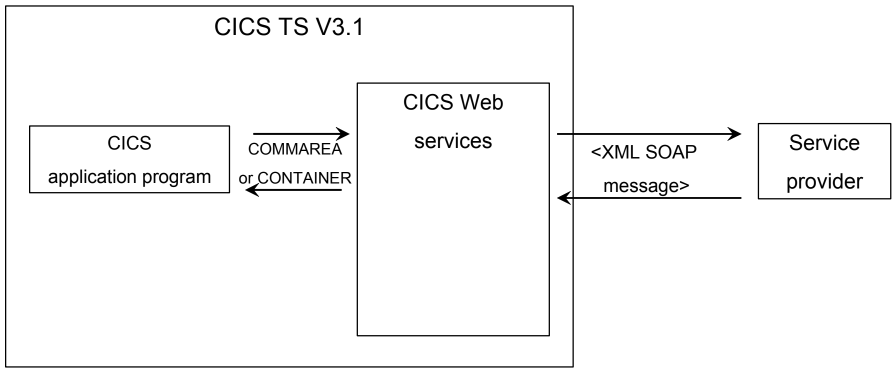


Figure 3-3   CICS as a service requester

When CICS is in the role of service requester, it must perform the following operations:

1. Build a request using data provided by the application program.
2. Send the request to the service provider.
3. Receive a response from the service provider.
4. Examine the response, and extract the contents that are relevant to the original application program.
5. Return control to the application program.

Note: Local optimization is possible when a CICS service requester invokes a CICS service provider application (see section 3.4.1, “Local optimization” on page 44).

3.4 Processing the outbound service request

Figure 3-4 on page 44 shows the processing that occurs when a service requester running in a CICS TS V3.1 region sends a SOAP message to a service provider.

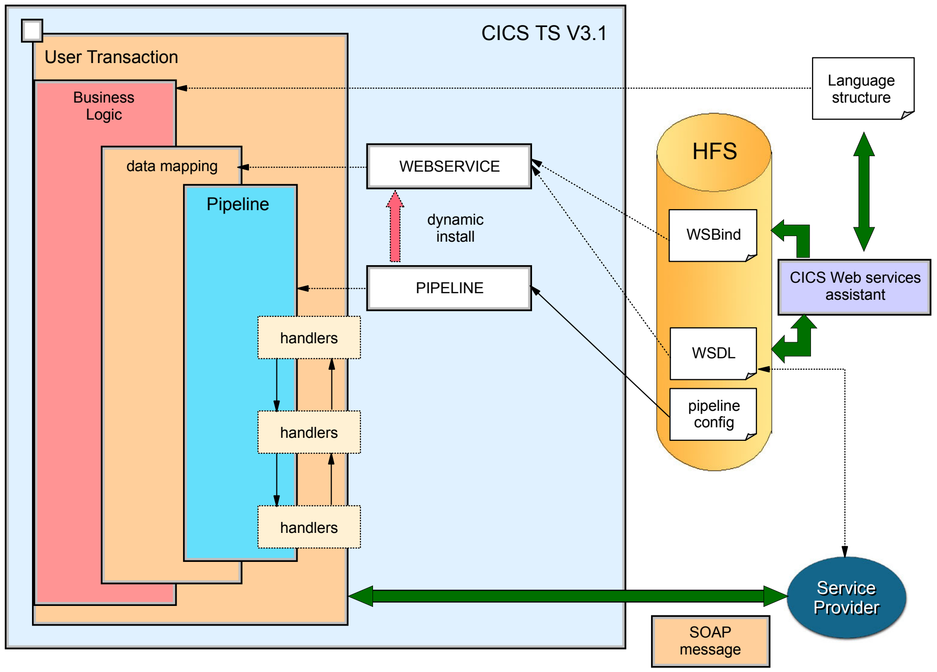


Figure 3-4 Outbound request processing

When the service requester issues the **EXEC CICS INVOKE WEBSERVICE** command, CICS uses the information found in the wsbind file that is associated with the specified WEBSERVICE definition to convert the language structure into an XML document. CICS then invokes the message handlers specified in the pipeline configuration file, and they convert the XML document into a SOAP message.

CICS sends the SOAP request message to the remote service provider via either HTTP or WebSphere MQ.

When the SOAP response message is received, CICS passes it back through the pipeline. The message handlers extract the SOAP body from the SOAP envelope, and the data mapping function converts the XML in the SOAP body into a language structure, which is passed to the application program in container DFHWS-DATA.

3.4.1 Local optimization

A special "local" optimization is possible when CICS is in the role of *both* service requester *and* service provider. In this case, CICS avoids the overhead of converting a language structure into an XML document by simply converting the **`EXEC CICS INVOKE WEBSERVICE`** command into an **`EXEC CICS LINK`** command.

Important: Invoking a CICS Web service using local optimization results in a significant performance benefit.

When an **`EXEC CICS INVOKE WEBSERVICE`** command is used to invoke a CICS service provider application, the provider application name in the Web service binding file associated with the WEBSERVICE resource is used to enable the local optimization of the Web service request. If you use this optimization, the request is optimized to an EXEC CICS LINK command as in Figure 3-5.

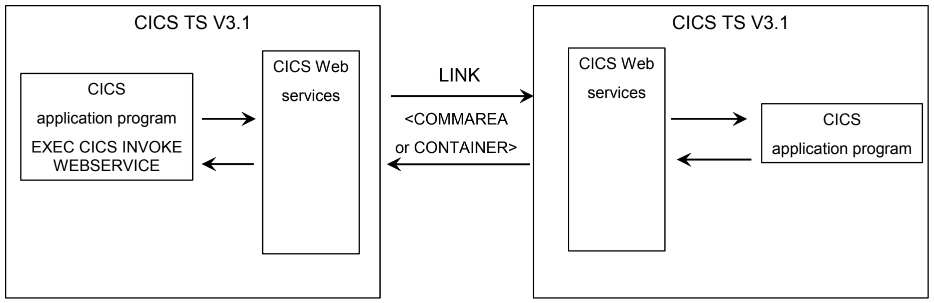


Figure 3-5 Invoking a CICS Web service using local optimization

The CICS service requester and service provider applications can be installed in the same CICS region or different regions. If they are in different regions, then an MRO or ISC connection must exist, which enables the LINK request to be shipped to the remote CICS region hosting the service provider application.

Note that this optimization has an effect on the behavior of the **`EXEC CICS INVOKE WEBSERVICE`** command when the Web service is not expected to send a response:

- When the optimization is not in effect, control returns from the **`EXEC CICS INVOKE WEBSERVICE`** command as soon as the request message is sent.
- When the optimization is in effect, control returns from the **`EXEC CICS INVOKE WEBSERVICE`** command only when the target program terminates.

When the Web service is expected to send a response, control returns from the command when the response is available.

Restriction: You can use this optimization only if the service provider application and the service requester application are deployed with the Web services assistant.

3.5 CICS resources for Web services

We now look in more detail at what CICS resources a systems programmer must implement in order to enable Web services in a CICS environment.

3.5.1 URIMAP

The URIMAP resource definition defines one of three different Web-related facilities in CICS. It is the value of the USAGE attribute on a URIMAP definition that determines which of the three facilities that particular definition controls.

1. Requests from a Web client to CICS as an HTTP server

   URIMAP definitions for requests for CICS as an HTTP server have a USAGE attribute of SERVER. These URIMAP definitions match the URLs of HTTP requests that CICS expects to receive from a Web client, and they define how CICS should provide a response to each request. You can use a URIMAP definition to tell CICS to do the following:

   – Provide a *static* response to the HTTP request, using a document template or z/OS UNIX® System Services HFS file

   – Provide a *dynamic* response to the HTTP request, using an application program that issues EXEC CICS WEB application programming interface commands

   – Redirect the request to another server, either temporarily or permanently

   For CICS as an HTTP server, URIMAP definitions incorporate most of the functions that were previously provided by the analyzer program specified on the TCPIPSERVICE definition. An analyzer program may still be involved if the processing path is required.

2. Requests to a server from CICS as an HTTP client

   URIMAP definitions for requests from CICS as an HTTP client have a USAGE attribute of CLIENT. These URIMAP definitions specify URLs that are used when a user application, acting as a Web client, makes a request through CICS Web support to an HTTP server. Setting up a URIMAP definition for this purpose means that you can avoid identifying the URL in your application program.

3. Web service requests

   URIMAP definitions for Web service requests have a USAGE attribute of PIPELINE. These URIMAP definitions associate a URI for an inbound Web service request—a request by which a client invokes a Web service in CICS—with a PIPELINE or WEBSERVICE resource that specifies the processing to be performed.

You can use a URIMAP with a USAGE attribute of PIPELINE to specify the following:

– The name of the transaction that CICS uses for running the pipeline alias transaction (the default is CPIH)

– The user ID under which the pipeline alias transaction runs

Figure 3-6 illustrates the purpose of a URIMAP resource definition for Web service requests.

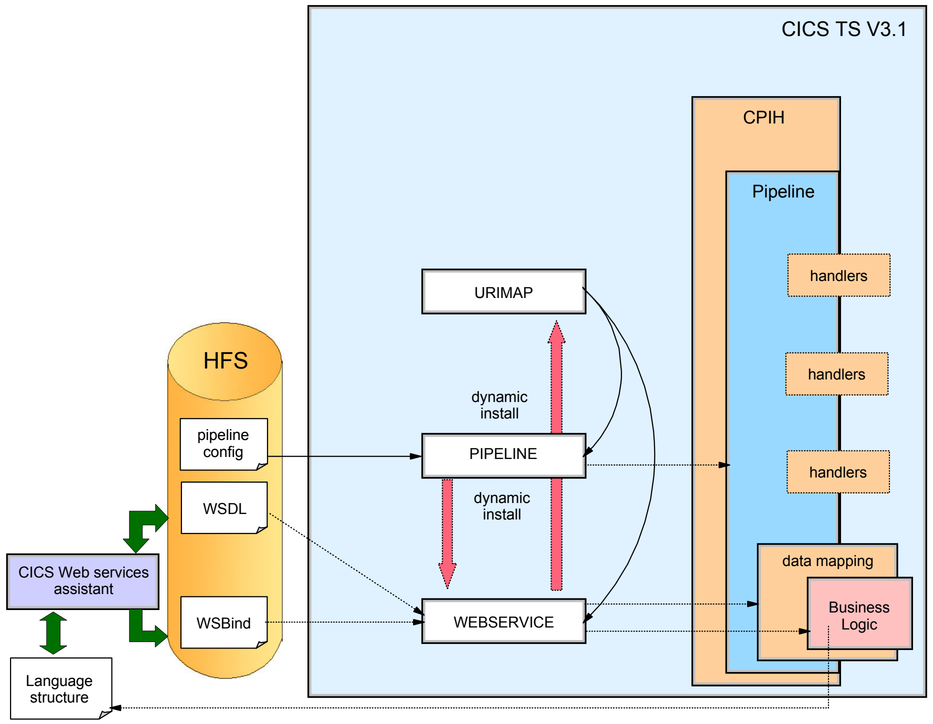


Figure 3-6 URIMAP relationships

You can create URIMAP resource definitions in the following ways:

- Use the CEDA transaction.
- Use the DFHCSDUP batch utility.
- Use CICSPlex SM Business Application Services.
- Use the EXEC CICS CREATE URIMAP command.

When you install a PIPELINE resource, or when you issue a **PERFORM PIPELINE SCAN** command (using CEMT or the CICS system programming interface), CICS scans the directory specified in the PIPELINE's WSDIR attribute (the pickup directory), and creates URIMAP and WEBSERVICE resources dynamically. For each Web service binding file in the directory, that is for each file with the wsbind suffix, CICS installs a WEBSERVICE and a URIMAP if one does not already exist. Existing resources are replaced if the information in the binding file is newer than the existing resources.

3.5.2 PIPELINE

A PIPELINE resource definition provides information about the message handlers that will act on a service request and on the response. The information about the message handlers is supplied indirectly. The PIPELINE definition specifies the name of an HFS file, called the pipeline configuration file, which contains an XML description of the message handlers and their configuration.

The most important attributes of the PIPELINE definition are as follows:

- WSDIR

  The WSDIR attribute specifies the name of the Web service binding directory (also known as the pickup directory). The Web service binding directory contains Web service binding files that are associated with the PIPELINE, and that are to be installed automatically by the CICS scanning mechanism. When the PIPELINE definition is installed, CICS scans the directory and automatically installs any Web service binding files it finds there.

  If you specify a value for the WSDIR attribute, it must refer to a valid HFS directory to which the CICS region has at least read access. If this is not the case, any attempt to install the PIPELINE resource will fail.

  If you do not specify a value for WSDIR, no automatic scan takes place on installation of the PIPELINE, and **`PERFORM PIPELINE SCAN`** commands will fail.

- SHELF

  The SHELF attribute specifies the name of an HFS directory where CICS will copy information about installed Web services. CICS regions into which the PIPELINE definition is installed must have full permission to the shelf directory: read, write, and the ability to create subdirectories.

  A single shelf directory may be shared by multiple CICS regions and by multiple PIPELINE definitions. Within a shelf directory, each CICS region uses a separate subdirectory to keep its files separate from those of other CICS regions. Within each region's directory, each PIPELINE uses a separate subdirectory.

  After a CICS region performs a cold or initial start, it deletes its subdirectories from the shelf before trying to use the shelf.

- CONFIGFILE

  This attribute specifies the name of the PIPELINE configuration file.

Figure 3-7 illustrates the purpose of the PIPELINE resource definition.

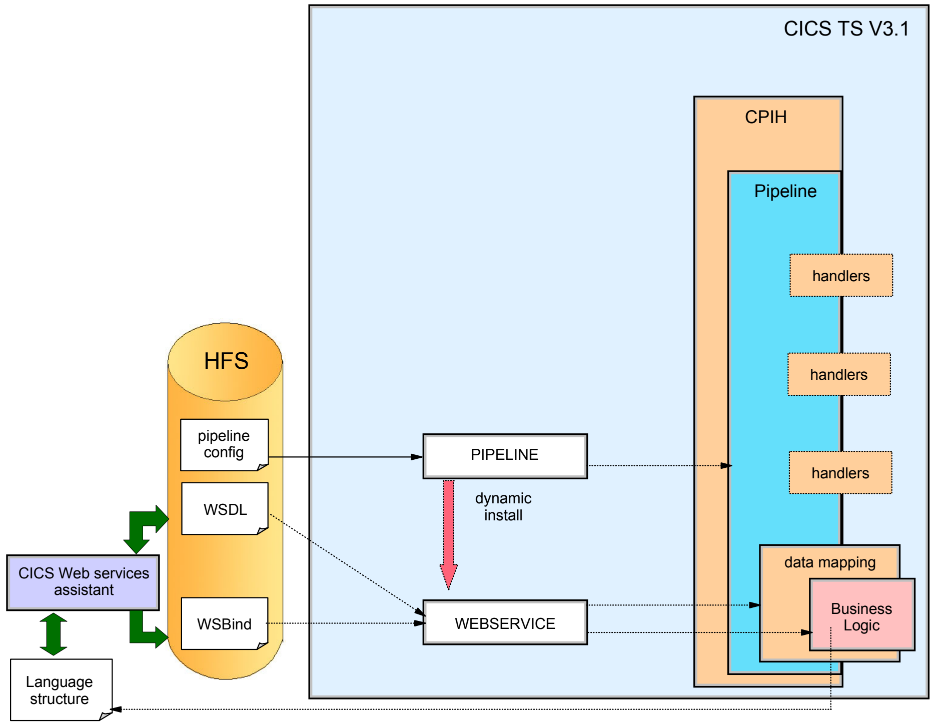


Figure 3-7 The PIPELINE resource relationships

You can create PIPELINE resource definitions in the following ways:

- Use the CEDA transaction
- Use the DFHCSDUP batch utility
- Use CICSPlex SM Business Application Services
- Use the EXEC CICS CREATE PIPELINE command

Pipeline configuration file

When CICS processes a Web service request, it uses a pipeline of one or more message handlers to handle the request. The configuration of a pipeline used to handle a Web service request is specified in an XML document, which is known as a *pipeline configuration file*. Use a suitable XML editor or text editor to work with your pipeline configuration files. The exact configuration of the pipeline will depend upon the specific needs of the application.

There are two kinds of pipeline configuration files:

- One describes the configuration of a service provider pipeline
- The other describes the configuration of a service requester pipeline

Each is defined by its own schema, and each has a different root element. The root element for a provider pipeline is `<provider_pipeline>`, while the root element for a requester pipeline is `<requester_pipeline>`.

The immediate child elements of the `<provider_pipeline>` element are as follows:

- A mandatory `<service>` element, which specifies the message handlers that are invoked for every request, including the terminal message handler. The terminal message handler is the last handler in the pipeline.
- An optional `<transport>` element, which specifies message handlers that are selected at run time based upon the resources that are being used for the message transport. For example, for the HTTP transport, you can specify that CICS should invoke the message handler only when the port on which the request was received is defined on a specific TCPIPSERVICE definition. For the WebSphere MQ transport, you can specify that CICS should invoke the message handler only when the inbound message arrives at a specific message queue.
- An optional `<apphandler>` element, which specifies the name of the program that the terminal message handler will link to by default, that is, the name of the target application program (or wrapper program) that provides the service. Message handlers can specify a different program at run time by using the DFHWS-APPHANDLER container, so the name coded here is not always the program to which it is linked.

  Important: When you use DFHLS2WS or DFHWS2LS to deploy your service provider, you must specify DFHPITP as the target program. DFHPITP will get the name of your target application program (or wrapper program) from the wsbind file.

  The `<apphandler>` element is used when the last message handler in the pipeline (the terminal handler) is one of the CICS-supplied SOAP message handlers.

  If you do not code an `<apphandler>` element, one of the message handlers *must* use the DFHWS-APPHANDLER container to specify the name of the program.

- An optional `<service_parameter_list>` element, which contains parameters that CICS makes available to the message handlers in the pipeline via container DFH-SERVICEPLIST.

Example 3-1 shows the sample service provider pipeline configuration file basicsoap11provider.xml.

Example 3-1 Configuration file for service provider

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline
xmlns="http://www.ibm.com/software/htp/cics/pipeline"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ibm.com/software/htp/cics/pipeline
provider.xsd ">
  <service>
    <terminal_handler>
      <cics_soap_1.1_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

Important: A pipeline can be configured to support SOAP 1.1 or SOAP 1.2. Within your CICS system, you can have many pipelines, some of which support SOAP 1.1 and some of which support SOAP 1.2.

Following are the immediate sub-elements of a `<requester_pipeline>` element:

- An optional `<service>` element, which specifies the message handlers that are invoked for every request
- An optional `<transport>` element, which specifies message handlers that are selected at run time, based upon the resources that are being used for the message transport
- An optional `<service_parameter_list>` element, which contains parameters that CICS makes available to the message handlers in the pipeline via container DFH-SERVICEPLIST

Example 3-2 shows the sample service requester pipeline configuration file basicsoap11requester.xml.

Example 3-2 Configuration file for service requester

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
```

```
<requester_pipeline
xmlns="http://www.ibm.com/software/htp/cics/pipeline"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.ibm.com/software/htp/cics/pipeline
requester.xsd ">
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler/>
    </service_handler_list>
  </service>
</requester_pipeline>
```

3.5.3 WEBSERVICE

The following three objects define the execution environment that allows a CICS application program to operate as a Web service provider or a Web service requester:

- The Web service description
- The Web service binding file
- The pipeline

The following three objects are defined to CICS on the following attributes of the WEBSERVICE resource definition:

- WSDLFILE
- WSBIND
- PIPELINE

The WEBSERVICE definition has a fourth attribute, VALIDATION, which specifies whether full validation of SOAP messages against the corresponding schema in the Web service description should be performed at run time. VALIDATION(YES) ensures that all SOAP messages that are sent and received are valid XML with respect to the XML schema.

Important: Validation of a SOAP message against a schema incurs considerable processing overhead, and you should normally specify VALIDATION(NO) in a production environment.

If VALIDATION(NO) is specified, sufficient validation is performed to ensure that the message contains well-formed XML.

Figure 3-8 illustrates the purpose of the WEBSERVICE resource definition.

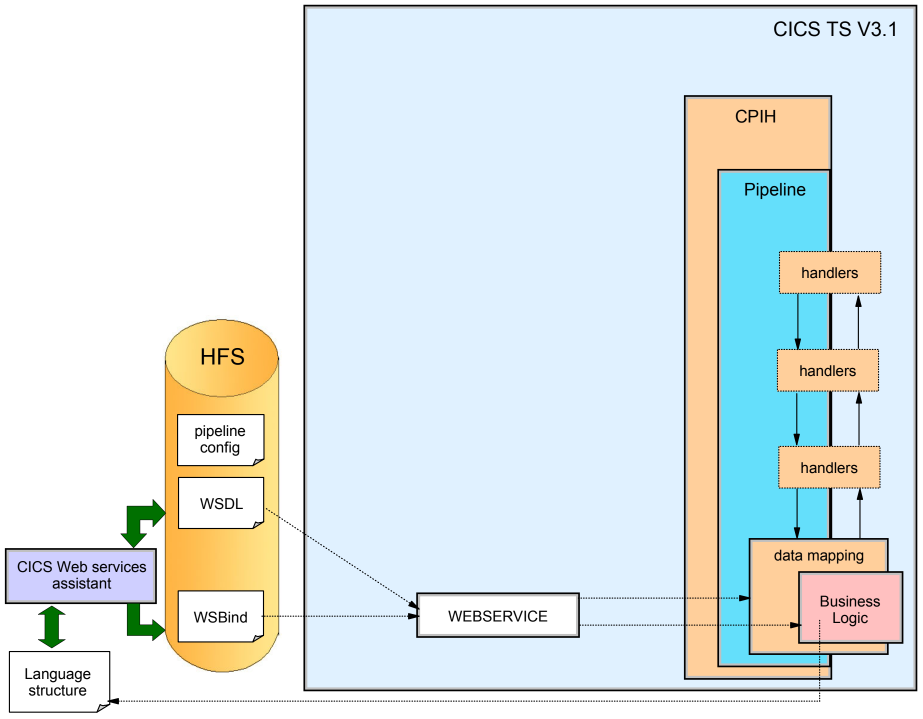


Figure 3-8 Webservice resource

You can create WEBSERVICE resource definitions in the following ways:

- Using the CEDA transaction
- Using the DFHCSDUP batch utility
- Using CICSPlex SM Business Application Services
- Using the **EXEC CICS CREATE WEBSERVICE** command

When you install a PIPELINE resource or when you issue a **PERFORM PIPELINE SCAN** command (using CEMT or the CICS system programming interface), CICS scans the directory specified in the PIPELINE's WSDIR attribute (the pickup directory) and creates URIMAP and WEBSERVICE resources dynamically. For each Web service binding file in the directory—each file with the wsbind suffix—CICS installs a WEBSERVICE and a URIMAP if one does not already exist. Existing resources are replaced if the information in the binding file is newer than the existing resources.

The CEMT INQUIRE WEBSERVICE command obtains information about a WEBSERVICE resource definition. The data returned depends on the type of Web service.

Web service binding file

A Web services description contains abstract representations of the input and output messages used by the service. When a service provider or service requester application executes, CICS needs information about how the content of the messages maps to the data structures used by the application. This information is held in a Web service binding file.

Web services binding files are created in the following manners:

- By utility program DFHWS2LS when language structures are generated from WSDL
- By utility program DFHLS2WS when WSDL is generated from a language structure

At run time, CICS uses information in the Web service binding file to perform the mapping between application data structures and SOAP messages.

3.5.4 TCPIPSERVICE

A TCPIPSERVICE definition is required in a service provider that uses the HTTP transport and contains information about the port on which inbound requests are received.

You can create TCPIPSERVICE resource definitions in the following ways:

- Using the CEDA transaction
- Using the DFHCSDUP batch utility
- Using CICSPlex SM Business Application Services
- Using the EXEC CICS CREATE TCPIPSERVICE command

3.5.5 Resources checklist

Figure 3-9 on page 55 shows the relationships between CICS Web services definitions.

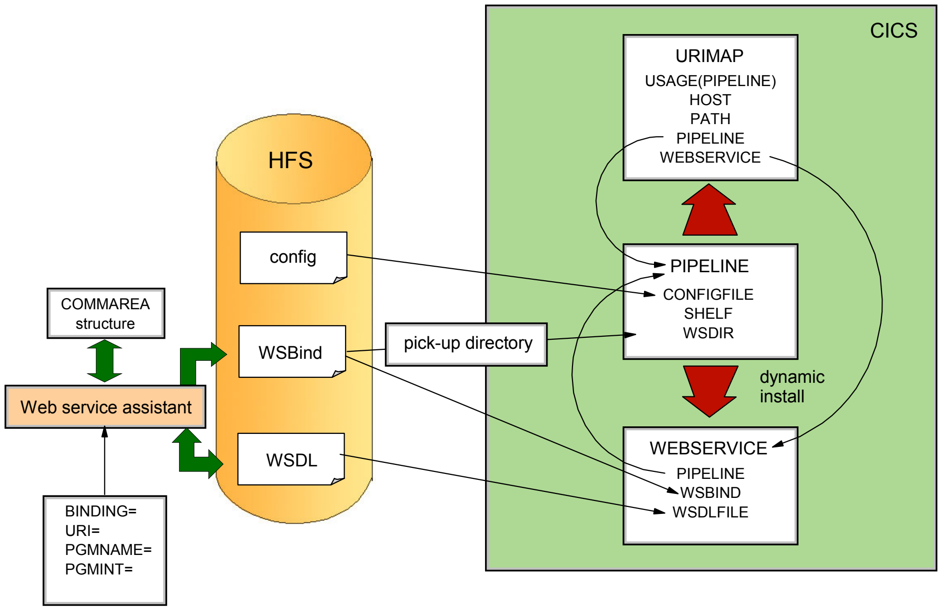


Figure 3-9 CICS Resource relationships

The resources that are required to support a particular application program depend upon the following:

- Whether the application program is a service provide or a service requester
- Whether the application is deployed with the CICS Web services assistant, or you write your own code to map between your application data and SOAP messages

Table 3-1 on page 56 is a checklist of resource definitions.

Table 3-1 Resource checklist

Service requester or provider	**CICS Web services assistant used**	**PIPELINE required**	**WEBSERVICE required**	**URIMAP required**	**TCPIPSERVICE required**
Provider	yes	yes	yes (1)	yes (1)	(2)
	no	yes	no	yes	(2)
Requester	yes	yes	yes	no	no
	no	yes	no	no	no

(1). When the CICS Web services assistant is used to deploy an application program, the WEBSERVICE and URIMAP resources can be created automatically when the PIPELINE's pickup directory is scanned. This happens when the PIPELINE resource is installed or as a result of a PERFORM PIPELINE SCAN command.
(2). A TCPIPSERVICE resource is required when the HTTP transport is used. When the WebSphere MQ transport is used, you must define a queue.

4

Modern Web services development tools

In this chapter we provide a brief overview of the main tools we use for developing CICS Web services. CICS TS3.1 introduces the Web services assistant, which is a set of batch utilities to aid in the generation of the files and resources needed for converting existing structures in CICS to Web service artifacts.

With reference to modern tooling, we also provide a brief description of the Eclipse platform and architecture and introduce one implementation of this framework—WebSphere Developer for System z. We describe the main features of this tool and define the general concepts of getting started and working in the environment, showing how powerful it is in the context of application development.

4.1 Web services assistant in CICS TS 3.1

The CICS Web services assistant is a set of batch utilities that can help you transform existing CICS applications into Web services and enable CICS applications to use Web services provided by external providers. The assistant supports rapid deployment of CICS applications for use in service providers and service requesters, with minimal programming effort.

When you use the Web Services Assistant for CICS, you do not have to write your own code for parsing inbound messages and for constructing outbound messages; instead, CICS maps data between the body of a SOAP message and the application program's data structure.

Resource definitions are, for the most part, generated and installed automatically. You do have to define PIPELINE resources, but you can, in many cases, use one of the pipeline configuration files that CICS provides.

The assistant can create a WSDL document from a simple language structure or a language structure from an existing WSDL document, and the assistant supports COBOL, C/C++, and PL/I. It also generates information used to enable automatic run-time conversion of the SOAP messages to containers and COMMAREAs, and vice versa.

However, the assistant cannot deal with every possibility, and there are times when you will need to take a different approach. Following are some examples:

- You do not want to use SOAP messages.

  If you prefer to use a non-SOAP protocol for your messages, you can do so. However, your application programs are responsible for parsing inbound messages and constructing outbound messages.

- You want to use SOAP messages, but do not want CICS to parse them.

  For an inbound message, the assistant maps the SOAP body to an application data structure. In some applications, you may want to parse the SOAP body yourself.

- The CICS Web services assistant does not support your application's data structure.

  Although the CICS Web services assistant supports the most common data types and structures, there are some that are not supported. For example, OCCURS DEPENDING ON and REDEFINES on data description entries are not supported. For full details on the data types and structures supported by the CICS Web services assistant, see the *CICS Web Services Guide,* SC34-6458.

In this situation, you should consider one of the following alternatives:

– Provide a wrapper program that maps your application’s data to a format that the assistant can support.

– Use WebSphere Developer for System z. See section 4.2, “WebSphere Developer for System z” on page 61.

4.1.1 Web services assistant utility programs

The CICS Web services assistant provides two utility programs: DFHLS2WS and DFHWS2LS. They are described in detail in this section.

DFHLS2WS

The DFHLS2WS program generates a Web service description and Web service binding file from a language structure. Example 4-1 shows sample JCL for running DFHLS2WS.

Example 4-1 DFHLS2WS JCL sample

```
//LS2WS JOB ’accounting information’,name,MSGCLASS=A
// SET QT=’’’’
//JAVAPROG EXEC DFHLS2WS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.CICS.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
LANG=COBOL
PGMNAME=DFH0XCMN
URI=exampleApp/inquireSingle
PGMINT=COMMAREA
WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind
WSDL=/u/exampleapp/wsdl/inquireSingle.wsdl
/*
```

The main input parameters are as follows:

- **PDSLIB**

  Specifies the name of the partitioned data set that contains the high-level language data structures to be processed.

- **REQMEM**

  Specifies the name of the partitioned data set member that contains the high-level language structure for the Web service request.

  – For a service provider, the Web service request is the input to the application program.
  – For a service requester, the Web service request is the output from the application program.

- **RESPMEM**

  Specifies the name of the partitioned data set member that contains the high-level language structure for the Web service response:

  – For a service provider, the Web service response is the output from the application program.
  – For a service requester, the Web service response is the input to the application program.

- **LANG**

  Specifies the language of the language structure to be created.

- **PGMNAME**

  Specifies the name of the target CICS application program that is being exposed as a Web service.

- **URI**

  In a service provider, this parameter specifies the relative URI that a client will use to access the Web service. CICS uses the value specified when it generates a URIMAP resource from the Web service binding file created by DFHLS2WS. The parameter specifies the path component of the URI to which the URIMAP definition applies.

- **PGMINT**

  For a service provider, specifies how CICS passes data to the target application program (using a COMMAREA or a channel).

- **WSBIND**

  Specifies the HFS name of the Web service binding file.

- **WSDL**

  Specifies the HFS name of the Web service description file.

In Chapter 6 we show how we used this utility to convert our sample application into Web service objects.

DFHWS2LS

DFHWS2LS generates a language structure and Web services binding file from a Web services description. Example 4-2 shows sample JCL for running DFHWS2LS.

Example 4-2 DFHWS2LS JCL sample

```
//WS2LS JOB 'accounting information',name,MSGCLASS=A
// SET QT=''''
//JAVAPROG EXEC DFHWS2LS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.CICS.SDFHSAMP
REQMEM=CPYBK1
RESPMEM=CPYBK2
LANG=COBOL
PGMNAME=DFHOXCMN
URI=exampleApp/inquireSingle
PGMINT=COMMAREA
WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind
WSDL=/u/exampleapp/wsdl/inquireSingle.wsdl
/*
```

4.2 WebSphere Developer for System z

WebSphere Developer for System z is based on the IBM Rational® Software Development Platform and facilitates the development of both Java and z/OS-based applications. It includes capabilities that make traditional z/OS mainframe development, Web development, and integrated composite development faster and more efficient.

Of particular interest to this book, WebSphere Developer for System z (WD/z) contains tools that support the development of Web services and the XML enablement of new and existing CICS COBOL and C/C++ applications. We describe some of the features and the use of WD/z that assisted us in the development, testing, and deployment of our sample application.

The implementation of how we used the product with our sample application is explored and demonstrated later in this IBM Redbooks publication.

4.2.1 Introducing WebSphere Developer for System z

WebSphere Developer for System z V7.0 comprises an interactive workstation-based environment and an integrated set of tools to help create, maintain, and re-use applications for traditional processing or inclusion in an SOA. It consists of a common workbench that supports end-to-end, model-based development, run-time testing, and rapid deployment of simple and complex applications.

Following are some of the main features of WD/z:

- Accelerates the development of Web applications, traditional COBOL and PL/I applications, Web services, XML-based interfaces, and high-level Enterprise Generation Language
- Includes significant enhancements that increase efficiency of traditional mainframe, Web development, and services
- Promotes the re-use and transformation of existing applications to help reduce costs and shorten the development cycle
- Facilitates mainframe application development, Web development, and integrated mixed workload or composite development faster and more efficient
- Enables business service integration for CICS applications
- Consists of workbench and integrated tools that support model-based application development, runtime testing, and rapid deployment
- Provides an interactive, workstation-based environment with convenient and consistent access to IBM z/OS datasets, HFS, and JES Spool files

In this book, we assume that the audience is a traditional mainframe developer with little experience in modern tooling techniques and terminologies.

This chapter further discusses some of the basics for WebSphere Developer for System z and defines the fundamental concepts upon which this product is built. We also provide some panel shots to assist with the visualization of the elements that make up the working environment of this tool.

4.2.2 The Eclipse platform

WebSphere Developer for System z is a product based on the Eclipse platform. The Eclipse platform is a GUI toolkit and plug-in architecture for building rich client applications. It is defined at *www.eclipse.org* as follows:

> *"Eclipse is an open source community whose projects are focused on building an open development platform comprised of extensible*

frameworks, tools and runtimes for building, deploying and managing software across the lifecycle."

Eclipse overview

The Eclipse platform incorporates the concept of a "work space" that maintains everything needed by the developer for building and testing a project. The work space is locally maintained on a developer's own workstation, containing the configuration settings of tools, plug-ins, the data objects being edited, and the intermediate and complete form of the components being developed.

The use of a local work space allows for very efficient team collaboration through repositories that can be Internet, rather than LAN, accessible.

Eclipse architecture

The Eclipse platform is a framework with a powerful set of services structured as subsystems that are implemented in one or more plug-ins. Following are the subsystems, or major components, that make up the framework:

- Platform runtime
- Work space
- Workbench
- Team support
- Help

Figure 4-1 shows a simplified view of the major components.

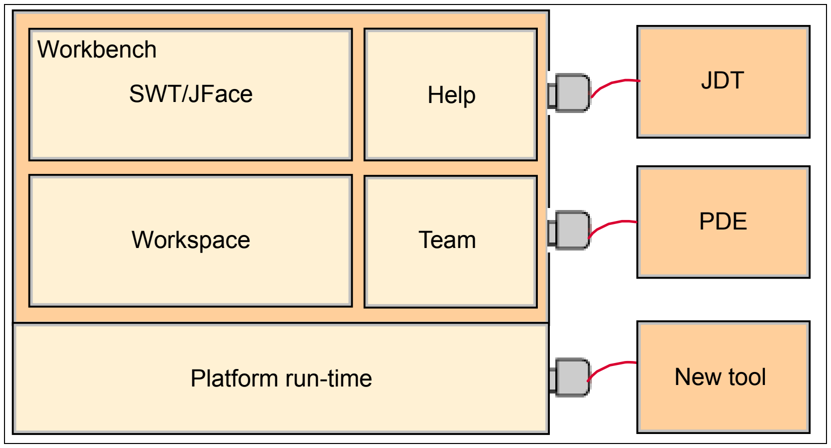


Figure 4-1 The Eclipse platform components

- **Platform**

  The platform run time is the kernel that discovers at start-up what plug-ins are installed and creates a registry of information about them. To reduce start-up time and resource usage, it does not load any plug-in until it is actually needed. Except for the kernel, everything else is implemented as a plug-in.

- **Work space**

  The work space is the plug-in responsible for managing the user's resources. This includes the projects the user creates, the files in those projects, and changes to files and other resources. The work space is also responsible for notifying other interested plug-ins about resource changes, such as files that are created, deleted, or changed.

- **Workbench**

  The workbench provides Eclipse with a user interface. It is built using the Standard Widget Toolkit (SWT)—a nonstandard alternative to Java's Swing/AWT GUI API—and a higher-level API, JFace, built on top of SWT that provides user interface components including file buffers, text handling, and text editors.

- **Team support**

  The team support component provides support for version control and configuration management. It adds views as necessary to allow the user to interact with what ever version control system (if any) is being used. Most plug-ins do not need to interact with the team support component unless they provide version control services.

- **Help**

  The help component parallels the extensibility of the Eclipse Platform itself. In the same way that plug-ins add functionality to Eclipse, help provides an add-on navigation structure that allows tools to add documentation in the form of HTML files.

4.2.3 The WebSphere Developer for System z Workbench

As we are working with the *WebSphere Developer for System z* (WD/z) product in this book, we now introduce some terms which are also found throughout other Eclipse-based products using WD/z as the visual example.

The term *workbench* refers to the desktop development environment. The workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of work space resources.

The user interfaces are based on editors, views, and perspectives. From a user's standpoint, a workbench window visually consists of several views and editors. Perspectives manifest themselves in the selection and arrangements of editors and views visible on the window. More than one workbench window usually shows on the desktop at any given time.

Editors allow the user to open, edit, and save objects and files. Views provide information about some object that the user is working within the workbench. A view may assist an editor by providing information about the document being edited.

When you first start up WD/z, a Welcome window appears, as in Figure 4-2. There are tutorials, samples, and overviews that are accessible from this window, which we highly recommend for first-time users to become familiar with the tooling.

To go to the workbench, click the highlighted icon, as shown in Figure 4-2.

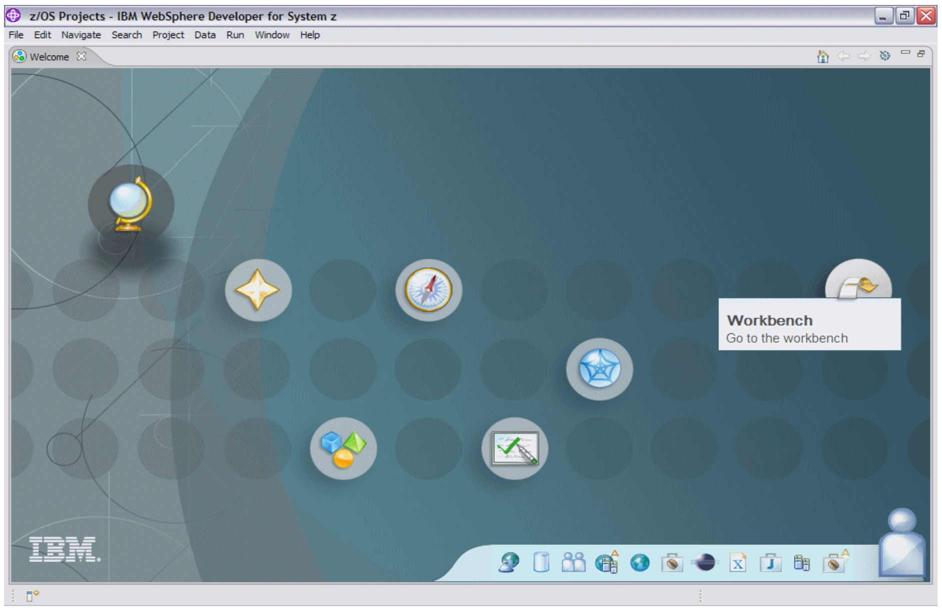
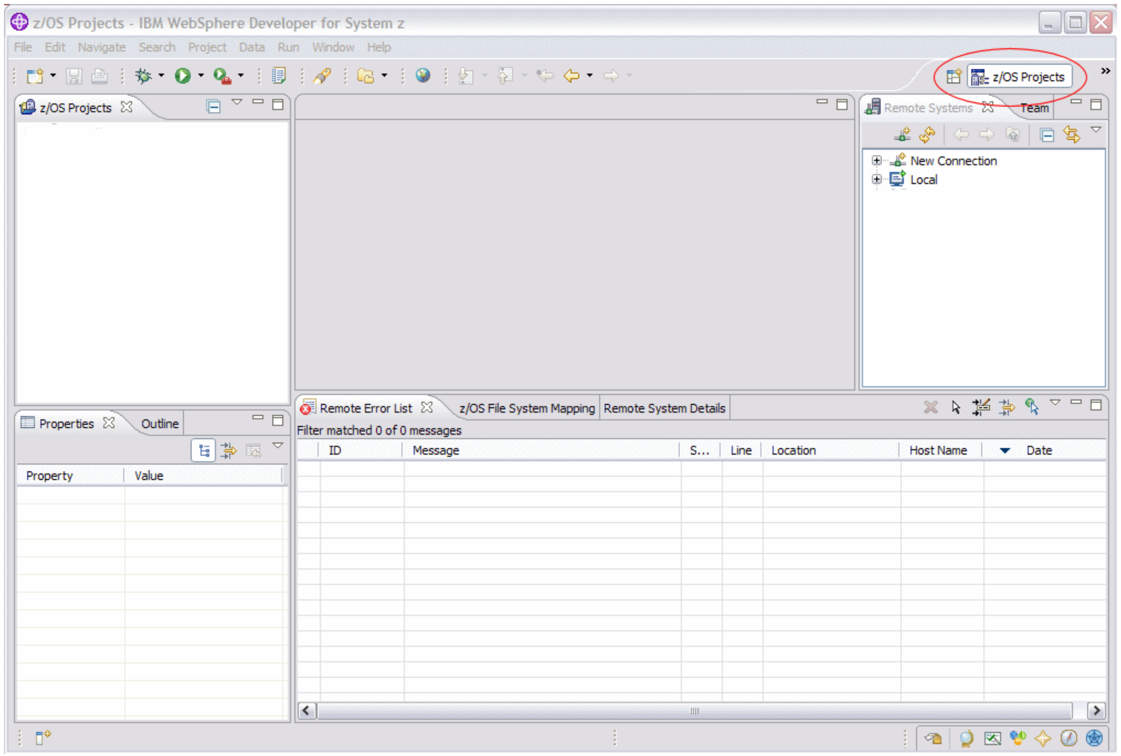

z/OS Projects - IBM WebSphere Developer for System z
File Edit Navigate Search Project Data Run Window Help
Welcome
Workbench
Go to the workbench
IBM

Figure 4-2 The WD/z Welcome window with Workbench selection

In WD/z, the default workbench comprises the z/OS Projects Perspective. This is shown in Figure 4-3 on page 66. Notice that the current perspective is set to z/OS Projects in the top-right of the workbench.

To change perspectives, there is a drop down selection icon to the left of the current perspective name. Alternatively, you can use **Window** → **Open Perspective** to select a different set of views that pertain to the type of development task you are performing.



Filter matched 0 of 0 messages

Figure 4-3 The z/OS Projects (default) Workbench

With reference to Figure 4-3, there are a number of views or panes that are now displayed.

The **Navigator** view in this perspective is labeled z/OS Projects, and it provides an Explorer-type view of projects and their members, which can be selected for edit, create, and delete. The greyed out panel to the right of the Navigator is the **Editor area**. Depending on the type of document selected in the Navigator, an appropriate editor window opens here, using a plug-in provided with the product. If an appropriate editor registered for a particular document type is not found (for example, a .doc file on a Windows® system), Eclipse will try to open the document using an external editor.

The **Outline** and the **Properties** views below the Navigator, present an outline of the document currently selected in the editor and the properties that are attributed to the file or document. The precise nature of these views depends on the editor and the type of document. For a Java source file, for example, the outline displays any declared classes, attributes, and methods.

The **Remote Systems** view allows connections to be made to Remote hosts and then provides an explorer view of the resources on the host. In WD/z, connections can be made to either AIX® or z/OS host systems.

We demonstrate connecting to a host and viewing remote resources in section 5.3.1, “Establish a connection to the host” on page 103.

Tip: All of the views and panes within the Workbench can be resized and dragged around the work space according to how you want to view the contents. Following are some other useful tips:

- Double-click the tab title of the pane to maximize the window, and double-click again to reduce back.
- Use **Window** → **Reset Perspective** to return to the default perspective.
- Other views can be added to the Workbench using **Window** → **Show View**.
- When editing a document or file, **ctrl+S** saves the changes; otherwise, if the window is closed, you are prompted to save the changes.

4.2.4 z/OS application development tools in WD/z

z/OS application development tools provide an interactive, workstation-based environment where you can develop mainframe applications in Assembler, COBOL, C/C++ or PL/I. The environment gives you an interactive way to edit on the workstation and prepare output on the mainframe. Interaction with z/OS allows you to do the following:

- Create or modify the code in the z/OS LPEX editor. The editor maintains fixed length records, sequence numbers, and file locking (ISPF ENQ/DEQ) as appropriate.
- Validate the source using the syntax check function.
- Debug the code.
- Generate and customize JCL as needed.

You can access z/OS data sets by way of a workstation-like directory structure, and you can process CLISTs and REXX™ EXECs in the following way:

1. Edit them on the workstation.
2. Transfer them to z/OS.

3. Run them on z/OS.
4. View the output in the workstation environment.

The code written using the editors in WD/z can target CICS, IMS™, DB2, or batch.

We will demonstrate further the z/OS application development tools described here in Chapters 5 and 6, where we use the tooling to assist in both the development and the testing of our sample application.

4.2.5 Web services development scenarios

WebSphere Developer for System z provides the development framework for major Web service enablement scenarios that are typical for a service-oriented architecture (SOA). The functionality comes with the Enterprise Service Tools (EST) in WD/z, which assist in creating enterprise applications that fit the established patterns of Web services enablement.

Following are the major Web services enablement scenarios that are typical for a service-oriented architecture:

- Bottom-up development
- Meet-in-the-middle development
- Top-down development

The scenarios are described in the following two contexts:

1. The creation of a Web service that invokes a single application. This context is applicable to most types of Enterprise Service Tools projects.
2. The creation of a comprehensive Web service capable of collecting and processing data from multiple CICS applications or from other Web services. This context is applicable to service flow projects.

In this book, we demonstrate the bottom-up approach within an EST single-service project.

Bottom-up development

This approach generates a Web service description and run-time specific inbound and outbound XML message processing from a high-level data structure. You can use this method to create a Web service provider program for an existing CICS application.

Meet-in-the-middle development

This defines mappings between high level data structures and WSDL, XML, or XSD files. You can use this method to generate run-time specific inbound and outbound XML message processing based on the mappings.

Top-down development

This generates a high-level language structure and run-time specific inbound and outbound XML message processing from a Web service description WSDL file. You can use this method to do the following:

- Create a Web service provider program (one that runs using Web services for CICS protocols) for a new application.
- Create a Web service provider program (one that runs using Web services for CICS protocols) for an existing application.
- Create a Web service requester program (one that runs using Web services for CICS protocols).

4.2.6 Enterprise Service Tools

Enterprise Service Tools (EST) is an integrated perspective that assists a CICS developer in the following tasks:

- Creating a CICS Web service that uses a new or existing CICS application as its application component.
- Creating a SOAP-enabled CICS Web service that uses a new or existing CICS application as its application component.
- Creating an IMS SOAP Gateway Web service that uses a new or existing IMS application as its application component.
- Creating a Web service that is not run-time specific from a new or existing application.
- Using the System z Database Application Generator to create a COBOL CICS application that accesses a Z/OS DB2 database.
- Developing a comprehensive Web service that collects and processes data from multiple sources, including CICS non-terminal applications, CICS terminal applications, and Web services.

Enterprise Service Tools combine capabilities that were formerly available in the **XML Services for the Enterprise** plug-in and the **Service Flow Modeller** plug-in. EST features an enhanced user interface that increases usability and simplifies the overall service and flow development process.

4.2.7 Web Services Enablement wizard

The Web Services Enablement wizard is a tool that supports the bottom-up approach for creating Web services based on existing CICS COBOL programs. It takes, as input, the COMMAREA copybook or a CONTAINER data structure. The XML structure and data types are then derived from the COBOL data declarations. Based on these, the Web Services Enablement wizard generates the set of artifacts shown in Figure 4-4.

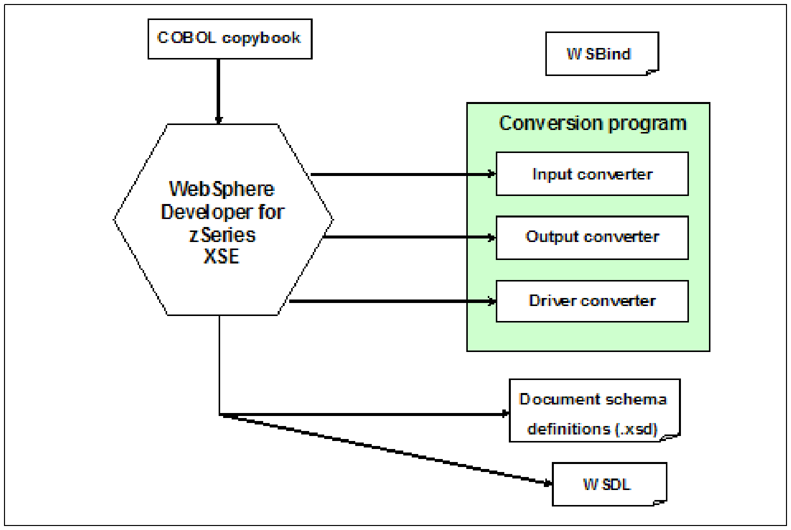


Figure 4-4 Artifacts generated by WD/z Web Services wizard

Following are the artifacts generated by the Web Services Enablement wizard:

Input converter

A COBOL program that takes an incoming XML document and maps it into the corresponding COBOL data structure that the existing CICS application expects.

Output converter

A COBOL program that takes the COBOL data results returned from the CICS application and maps them to an XML document.

Converter driver

A COBOL program that shows how the input and output converters can be used to interact with the existing CICS application.

Input document XML schema definition (XSD)

XML schema that describes the incoming XML document.

Output document XML schema definition (XSD)

XML schema that describes the outgoing XML document.

WSDL

Web service description file.

WSBind

Web service binding file.

For additional information visit the WebSphere Developer for System z Web site at the following Web address:

`http://www.ibm.com/software/awdtools/devzseries/`

We demonstrate the use of this wizard to generate the Web service artifacts for our sample program in section 6.2, “Using WD/z to generate WSDL” on page 123.

5

Development of the Change of Address CICS application

In this chapter we describe our Change of Address application CICSWSAP and how we developed the presentation logic and the business logic into separate functions or services that can be easily exposed as Web services.

Here we describe the development of the mainframe components of the Change of Address application. This includes the following:

- Breakdown of the CICS application
- Development of the BMS presentation logic
- Functions of the Business Logic components
- The MQ and Message Broker components

We also demonstrate how WebSphere Developer for System z can be used to develop presentation logic through the BMS Editor. We assume that you have very little experience with WD/z; therefore, some basic functions are described throughout this chapter to familiarize you with enough information so you can utilize some of the z/OS development functionality.

Full source code is downloadable via the Additional Materials link for this IBM Redbooks publication.

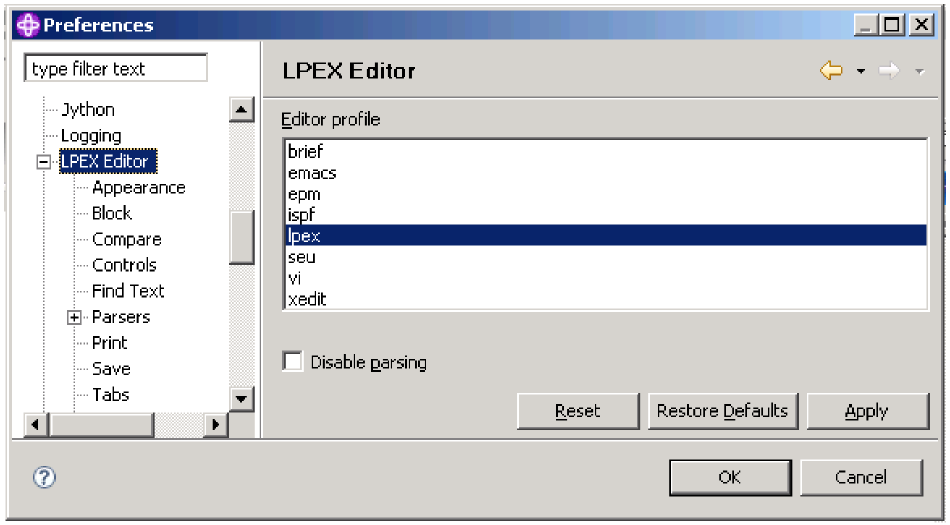
5.1 Breakdown of the CICS application

This section describes the details of the Change of Address application. An overview of the complete application was given in section 1.5, “The Change of Address application” on page 8, so we will concentrate on the CICS application specifically here.

5.1.1 Overview of the approach to CICS application development

The CICS application was initially developed as a CICS/BMS menu driven system. This enabled our developers, who were new to CICS Web services, to develop and test a traditional application while they learned the new WD/z tooling. By using careful application structuring, we kept the business logic well separated from the BMS presentation logic. This is the key to easy transition to a CICS Web service using the CICS Web Services Assistant.

As discussed in Chapter 4, “Modern Web services development tools” on page 57, we employed the newly released WebSphere Application Developer for Systems z V7 to create and edit the BMS maps. This facility is a great benefit to the traditional BMS developer as it provides true *drag and drop* BMS map creation. BMS maps are edited directly out of mainframe PDS(E) libraries. Likewise, JCL editing and job-submission is friendly and efficient. The *default* LPEX editor provides more of a Windows Notepad-like editor. We say *default* here because there is a significant number of editor styles available. If desired, an ISPF- like editor is available under **Window** → **Preferences**: **LPEX**. (See Figure 5-1 on page 75.) The editors also provide colored parsing and syntax error notifications as you type.



LPEX Editor

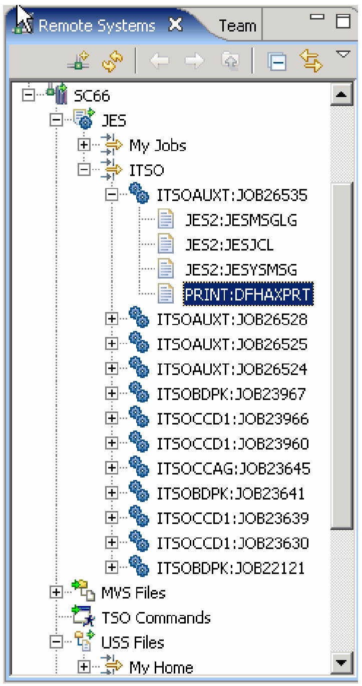
Editor profile

brief
emacs
epm
ispf
lpex
seu
vi
xedit

Disable parsing

Figure 5-1 LPEX Editor choices

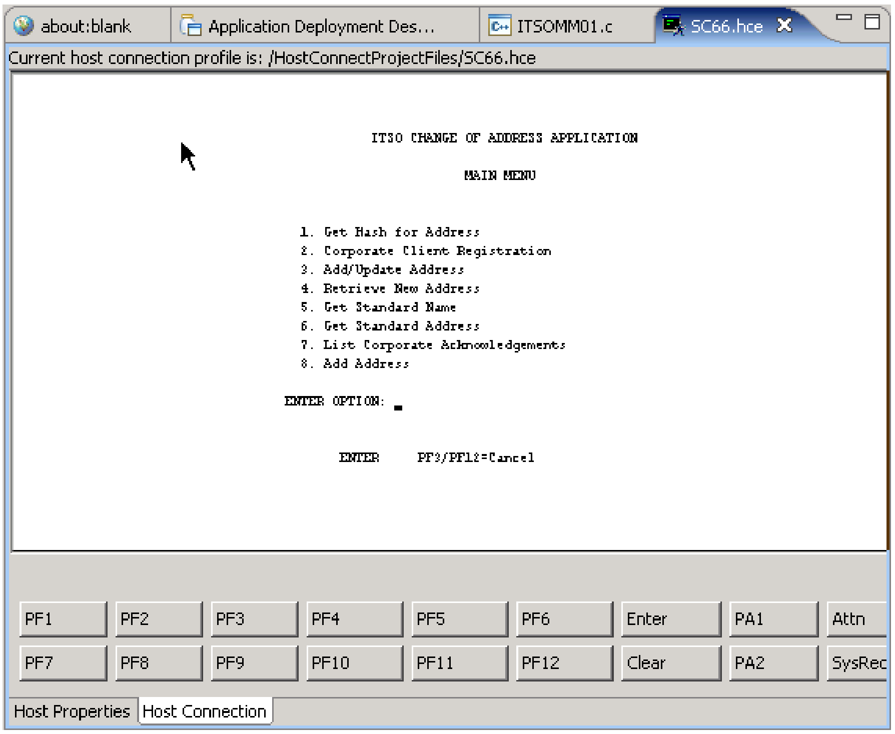
Job output is readily available under the JES key of the Remote Systems pane:



SC66
JES
My Jobs
ITSO
ITSOAUXT:JOB26535
JES2:JESMSGLG
JES2:JESJCL
JES2:JESYSMSG
PRINT:DFHAXPRT
ITSOAUXT:JOB26528
ITSOAUXT:JOB26525
ITSOAUXT:JOB26524
ITSOBDPK:JOB23967
ITSOCCD1:JOB23966
ITSOCCD1:JOB23960
ITSOCCAG:JOB23645
ITSOBDPK:JOB23641
ITSOCCD1:JOB23639
ITSOCCD1:JOB23630
ITSOBDPK:JOB22121
MVS Files
TSO Commands
USS Files
My Home

Figure 5-2 Accessing Job output

WD/z also contains a 3270 emulator, allowing full window-based testing of applications as they develop.



Current host connection profile is: /HostConnectProjectFiles/SC66.hce

```
                    ITSO CHANGE OF ADDRESS APPLICATION

                                MAIN MENU

          1. Get Hash for Address
          2. Corporate Client Registration
          3. Add/Update Address
          4. Retrieve New Address
          5. Get Standard Name
          6. Get Standard Address
          7. List Corporate Acknowledgements
          8. Add Address

        ENTER OPTION:


              ENTER      PF3/PF12=Cancel
```

Figure 5-3 A 3270 screen in WD/z

What this means is that the CICS developer is freed from the need to continually switch between 3270 mainframe sessions and WD/z in the edit-compile-test sequence.

During development, as simple functions completed testing as a BMS application, we created the WSDL & WSBind using the DFHLS2WS utility. We carried out simple testing of these generated Web services initially using the Web Services Explorer feature that is part of the underlying RAD software.

5.1.2 Separation of presentation, business, and data logic

Separation of business logic from presentation and data logic has become a tenet of modern programing. This stems from the 3-tier software design pattern that originates even before client-server first became popular. Presentation engines (clients) made calls to the business logic (server), which called a back end database system. The presentation layer never calls the data layer directly.

The data layer is typically insulated from the business layer by a standard interface such as SQL.

This is good practice for many reasons including the following:

- Applications can have a multitude of presentation interfaces. Perhaps a Web interface as well as a 3270 interface.
- Organizations tend to have graphic designers produce presentation interfaces, particularly for Web applications. This staff should not have to worry about business or data logic.
- Independence of presentation implementation from business or data implementation. Indeed, they might be on different platforms.
- Components can be upgraded independently.
- The usual *non-functional* reasons such as reliability, scalability, security, robustness, recoverability.

Keeping these three layers separate is easy when developing from the ground up. Keeping all data in a database system, then ensuring separation of the BMS calls from the business logic is then fairly straightforward.

Importantly, the interface to the business logic is then a mapped data structure of input and output parameters. For smaller structures (< 32Kb), we use CICS COMMAREAS to contain this structure. But the facility is present in CICS to handle larger structures using containers and channels. Given the simplicity of our application, we mostly use COMMAREAS.

5.1.3 Overview of the application

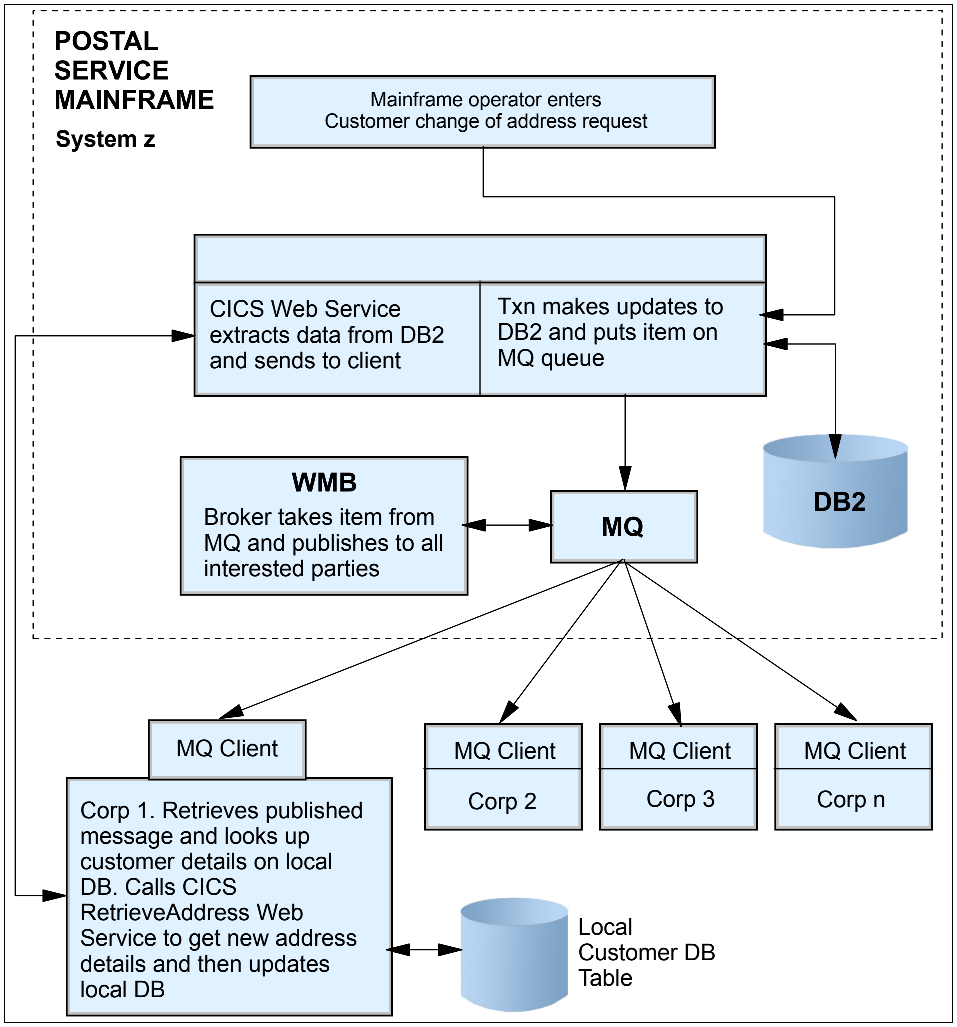

POSTAL SERVICE MAINFRAME
System z
Mainframe operator enters Customer change of address request
CICS Web Service extracts data from DB2 and sends to client
Txn makes updates to DB2 and puts item on MQ queue
WMB
Broker takes item from MQ and publishes to all interested parties
MQ
DB2
MQ Client
Corp 1. Retrieves published message and looks up customer details on local DB. Calls CICS RetrieveAddress Web Service to get new address details and then updates local DB
MQ Client
Corp 2
MQ Client
Corp 3
MQ Client
Corp n
Local Customer DB Table

Figure 5-4 Application design of CICSWSAP

5.1.4 Database schema

The following tables are defined on our DB2 for z/OS system:

```
ADDRESS (
ADDRESSHASH  INTEGER   PRIMARY KEY NOT NULL,
ADDRESSLINE1 CHAR(100) NOT NULL,
ADDRESSLINE2 CHAR(100),
```

SUBURB CHAR(50) NOT NULL,
STATE CHAR(10) NOT NULL,
POSTCODE CHAR(10) NOT NULL,
LASTUPDATE TIMESTAMP NOT NULL)

NAMES (
NAMEREF INTEGER   PRIMARY KEY NOT NULL GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1),
FIRSTNAME   CHAR(50) NOT NULL,
MIDDLENAME CHAR(50),
LASTNAME CHAR(50) NOT NULL,
ADDRESSHASH INTEGER NOT NULL,
OLDADDRESSHASH INTEGER,
LASTUPDATE TIMESTAMP NOT NULL)
CORPCLIENTS(
ID INTEGER   PRIMARY KEY NOT NULL
GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1),
SHORTNAME   CHAR(10) NOT NULL,
LONGNAME   CHAR(50) NOT NULL,
DESCRIPTION VARCHAR(500))

AUDIT (
AUDITREF   INTEGER   PRIMARY KEY NOT NULL
GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1),
ACTIVITYTIME TIMESTAMP,
ACTIVITY   CHAR(10) CHECK (ACTIVITY IN
('ADD', 'DELETE', 'UPDATE', 'READ', 'ACK')),
NAMEREF   INTEGER   NOT NULL,
ACTIVITYBY   INTEGER   NOT NULL)

Figure 5-5 on page 80 displays the relationships between these tables.

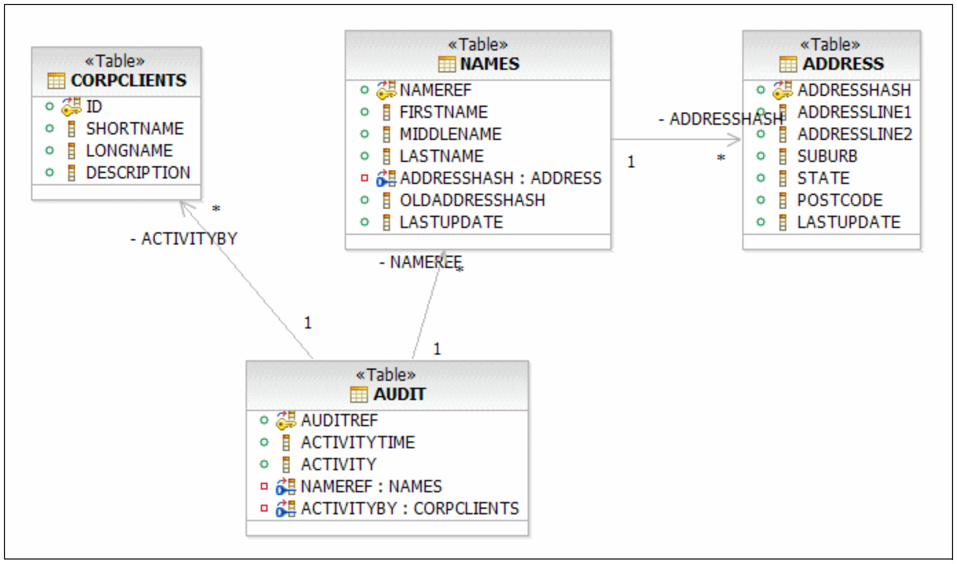


Figure 5-5   Database relationships

5.1.5 Application schema

This section explains the application schema.

Programs

Programs have the following name structure:

ITSOxxnn

Where: xx indicates the function, for example CA is Corporate Acknowledgements and nn is a two-digit number of the program.

Typically nn=01 is the initial program that displays the first BMS menu of the function. nn=02 retrieves the BMS map and sets up the COMMAREA to call the business function and then displays the results. nn=03, 04 ... are the business layer programs of this particular function.

Transactions

Transactions are identified as follows: **IT**nn

where nn=00, 01 ...

The only transaction worthy of individual mention is **IT00** - the main menu transaction.

Map sets/maps

Map sets are identified as follows: ITSOMSn

where n=0, 1 ...

WMQ queues

WMQ queues are prefixed with CICSWSAP. The main publication queue is called CICSWSAP.PUBLICATION.QUEUE, from which the pub/sub function (see Chapter 7) distributes messages to subscriber queues. These are typically prefixed CICSWSAP.ADDRESS.CHANGE.

5.1.6 Application functions

Following are the major business functions of our application. There are some other minor functions and test harnesses included in the source.

Get Hash

Description

Return a hash key calculated from the full address supplied in the input parameters. This key should be unique for the given parameters.

Availability

Publicly available as a Web service call.

Logic

Calculate hash value from string AddressLine1 + AddressLine2 + Suburb + State + Postcode.

Algorithm is known as **djb2**. Quick, efficient with a good spread of results.

Input parameters

Address input parameters must be in the standard format provided by the **StandardAddress** function in Table 5-1.

Table 5-1 GetHash (ITSOGH03) input parameters

Name	Format	Requirement	Comments
AddressLine1	Text	Required	In Standard Format
AddressLine2	Text	Optional	In Standard Format
Suburb	Text	Required	In Standard Format
State	Text	Required	In Standard Format

Name	Format	Requirement	Comments
PostCode	Text	Required	In Standard Format

Output parameters

Table 5-2 GetHash (ITSOGH03) Output Parameters

Name	Format	Comments
AddressHash	Large Integer	Up to 11 digits; possibly negative.
ReturnCode	Integer	
Reason	Text	

Return codes

RC=0 mean OK.

Only returns RC=0. Might fail if input parms are in read-protected storage.

Corporate client registration

Description

Register a corporate client. Note, this does not register a pub/sub subscription, although such functionality could easily be built into this program.

Availability

Internal only.

Input parameters

Table 5-3 CorpClient (ITSOCC03) input parameters

Name	Format	Requirement	Comments
ShortName	Text	Required	------------------------
LongName	Text	Optional	------------------------
Description	Text	Optional	------------------------

Output parameters

Table 5-4 CorpClient (ITSOCC03) output parameters

Name	Format	Comments
CorpClientId	Integer	----------------------------------

Name	Format	Comments
ReturnCode	Integer	----------------------------------
Reason	Text	---------------------------------

Return codes

RC=0 means OK. CorpClient Added.
RC=1 means SQL INSERT to CORPCLIENTS table failed. See Reason.
RC=2 means SQL SELECT to obtain CorpClient ID failed. See Reason.

Logic

- Check input parameter
- INSERT new client into CORPCLIENTS table
- If INSERT failed
  - Set RC=1, Reason=SQLCODE, SQLSTATE
  - Return
- SELECT row just entered to get generated CorpClient ID
- If SELECT failed
  - Set RC=2, Reason=SQLCODE, SQLSTATE
  - Return
- Set RC=0, CorpClientId, Reason
- Return

Add address

Description

Add an address to DB2 Address table or return an error code. This function is called internally by Add/Update Address only, but it is a significant function factored out of the Add/Update Address. For this reason, this program is called ITSO**UA**03, not ITSOAA03.

Availability

Internal only.

Input parameters

Table 5-5 AddAddress (ITSOUA03) input parameters

Name	Format	Requirement	Comments
AddressLine1	Text	Required	In standard format

Name	Format	Requirement	Comments
AddressLine2	Text	Optional	In standard format
Suburb	Text	Required	In standard format
State	Text	Required	In standard format
PostCode	Text	Required	In standard format

Output parameters

Table 5-6 AddAddress (ITSOUA03) output parameters

Name	Format	Comments
AddressHash	Large Integer	Up to 11 digits. Possibly negative.
ReturnCode	Integer	-------------------------------
Reason	Text	-------------------------------

Return Codes

RC=0 mean OK
RC=1 means Address Already exists
RC=2 means SQL Failure

Logic

- ▶ Check input parameters
- ▶ Call GetHash to get HashCode for address
- ▶ INSERT address into ADDRESS table
- ▶ If INSERT failed
  - – If SQLCode indicates address already exists
    - • Set RC=1, Reason=Address Already Exists
    - • Set AddressHash
    - • Return
  - – Otherwise there was an SQL Error
    - • Set RC=2, Reason=SQLCODE, SQLSTATE
    - • Set AddressHash=0
    - • Return

- INSERT was OK:
  - Set RC=0
  - Set AddressHash
  - Return

Add/update address

Description

Add or update the address of a relocatee in the local DB2 Names and Address tables. Post notification to broker of update.

Availability

Internal only.

Input parameters

Table 5-7 Add/update address (ITSOUA04) input parameters

Name	Format	Requirement	Comments
FirstName	Text	Required	In standard format
MiddleName	Text	Optional	In standard format
LastName	Text	Required	In standard format
AddressLine1	Text	Required	Of new address in standard format
AddressLine2	Text	Required	Of new address in standard format
Suburb	Text	Required	Of new address in standard format
State	Text	Required	Of new address in standard format
Postcode	Text	Required	Of new address in standard format
AddressHash	Integer	Required	Redundant now. Must set to 0
ClientId	Integer	Required	----------------------

Output Parameters

Table 5-8 Add/update address (ITSOUA04) Output Parameters

Name	Format	Comments
NameRef	Integer	-------------------------------
AddressHash	Large Integer	Up to 11 digits. Possibly negative.
ReturnCode	Integer	--------------------------------
Reason	Text	--------------------------------

Return Codes

RC=0 means OK
RC=1 means Insert Name Failed - SQL error
RC=2 means Name already exists when trying to do an Add
RC=3 means Current and new address are the same
RC=4 means call to AddAddress Failed
RC=5 means Supplied AddressHash didn't match existing address for name
RC=6 means Cannot update non-existent name entry
RC=7 means Notification to broker failed (MQ error)

Logic

- Attempt to retrieve NAMES table entry for the person named
- If there was a SQL error, other than no NAMES entry
  - Set RC=2 Reason=SQLCODE SQLSTATE
  - Return
- If there was no NAMES entry
  - Call AddAddress to add the address
  - If call failed (“Address already exists” is not a failure)
    - Set RC=4, Reason=AddAddress reason
    - return
  - Insert new entry into NAMES table with AddressHash just returned
  - If INSERT failed
    - Set RC=1, Reason= SQLCODE SQLSTATE
    - Perform rollback
    - Return
  - SELECT the row just added to retrieve the NAMEREF generated

  - If SELECT failed
    - Set NameRef = -1
    - Continue on. Not a big problem.
  - MQPUT AddressHash on to publication queue
    - If MQPUT failed
      Set RC=7, Reason= MQRC
      Perform rollback
      Return
- NAMES entry already exists
  - Call AddAddress to add the address
  - If call failed ("Address already exists" is not a failure)
    - Set RC=4, Reason=AddAddress reason
    - return
  - Update NAMES db entry using new address & hashcode
  - If UPDATE failed
    - Set RC=1, Reason= SQLCODE SQLSTATE
    - Perform rollback
    - Return
  - MQPUT old AddressHash on to publication queue
    - If MQPUT failed
      Set RC=7, Reason= MQRC
      Perform rollback
      Return
- Set RC=1, Reason=OK, AddressHash & NameRef

Standard name

Description

Format names in a standard way in terms of capitalization, spacing, and spelling. There are whole application suites in existence to perform these sorts of functions. This is not critical to our test application, but it would be in a real situation. There might even be public Web services in existence that already do this.

Availability

Publicly available as a Web service call.

Input parameters

Table 5-9 StandardName (ITSOSN03) input parameters

Name	Format	Requirement	Comments
FirstName	Text	Required	----------------------
MiddleName	Text	Optional	----------------------
LastName	Text	Required	----------------------

Output parameters

Table 5-10 StandardName (ITSOSN03) output parameters

Name	Format	Comments
FirstName	Text	In standard format
MiddleName	Text	In standard format
LastName	Text	In standard format
ReturnCode	Integer	------------------------------
Reason	Text	------------------------------

Return Codes

RC=0 means OK
RC=1 means Not OK

Logic

- Check input parameters
- Format Name

Standard address

Description

Format addresses in a standard way. This allows different organizations to compare addresses in standard fashion. There are whole application suites in existence to perform these sorts of functions. This is not critical to our test application but it would be in a real situation. There might even be public Web services in existence that already do this.

Availability

Publicly available as a Web service call.

Input parameters

Table 5-11 StandardAddress (ITSOSA03) input parameters

Name	Format	Requirement	Comments
AddressLine1	Text	Required	------------------------
AddressLine2	Text	Optional	-----------------------
Suburb	Text	Required	-----------------------
State	Text	Required	-----------------------
Postcode	Text	Required	-----------------------

Output parameters

Table 5-12 StandardAddress (ITSOSA03) output parameters

Name	Format	Comments
AddressLine1	Text	In standard format
AddressLine2	Text	In standard format
Suburb	Text	In standard format
State	Text	In standard format
Postcode	Text	In standard format
ReturnCode	Integer	-------------------------------
Reason	Text	-------------------------------

Return Codes

RC=0 means OK
RC=1 means Not OK

Logic

- Check input parameters
- Format name

List corporate acknowledgements

Description

Return a list of the corporate clients that were sent the new address of the relocatee. This does not mean they actually changed this in their own databases, but they should have.

Availability

Restricted availability. Should be available to relocatee perhaps by a secured Web page.

Input parameters

Table 5-13 CorpAcknowledgements (ITSOCA03) input parameters

Name	Format	Requirement	Comments
FirstName	Text	Required	In standard format
MiddleName	Text	Optional	In standard format
LastName	Text	Required	In standard format
AddressLine1	Text	Required	In standard format
AddressLine2	Text	Required	In standard format
Suburb	Text	Required	In standard format
State	Text	Required	In standard format
Postcode	Text	Required	In standard format

Output Parameters

Table 5-14 CorpAcknowledgements (ITSOCA03)output parameters

Name	Format	Comments
NumberOfResponses	Integer	--------------------------------
CorpClient1	Text	--------------------------------
:	Text	--------------------------------
CorpClientn	Text	--------------------------------
ReturnCode	Integer	--------------------------------
Reason	Text	--------------------------------

Return Codes

RC=0 means OK
RC=1 means Bad input parameter
RC=2 means SQL Error

Logic

- Validate input parms

- If parms invalid
  - Return RC=1 Reason="Bad Paramater: " + parm in error
- Query Audit table for all the corp-clients who were told of new address
- If query failed
  - Return RC=2 Reason="Query Failed: " + SQL Reason
- Otherwise
  - Return RC=0 Reason="OK" with results

5.2 Developing the presentation logic using the BMS Editor in WD/z

As part of our ground-up development of the sample application, we show how to use the BMS Editor in WD/z to create our BMS maps. This is a modern tooling alternative to using the traditional designer (such as SDF II) or straight coding of BMS source.

The WD/z BMS Editor is associated with .bms map set files and uses the following Eclipse views in its perspective:

- **Project Explorer**

  Lists the projects and the BMS map sets for that project
- **Outline**

  Lists the maps and artifacts within the map
- **Palette**

  Items associated with BMS maps are listed that can be easily dragged onto the design canvas
- **Properties**

  Lists the attributes for the maps and the map sets that are highlighted in the *Outline* view

This section demonstrates how we created one of the BMS maps for this project.

5.2.1 Create a Project

The following steps show how to create a project, into which we will be able to create the map set and map objects.

1. Start up WD/z.
2. Click **File** → **New** → **Project**.
3. Select **General** → **Project**.
4. Click **Next**.
5. Type the name of the project, in this example **Redbooks**.
6. Click **Finish**.

The local project can be seen in the *Project Explorer* view in the **Other Projects** group.

Use the following steps to view the Project Explorer pane:

1. Select **Window** → **Show View** → **Other**.
2. Expand the **General** folder.
3. Click Project Explorer.

The project Redbooks can be seen in the Project Explorer view, as seen in Figure 5-6.

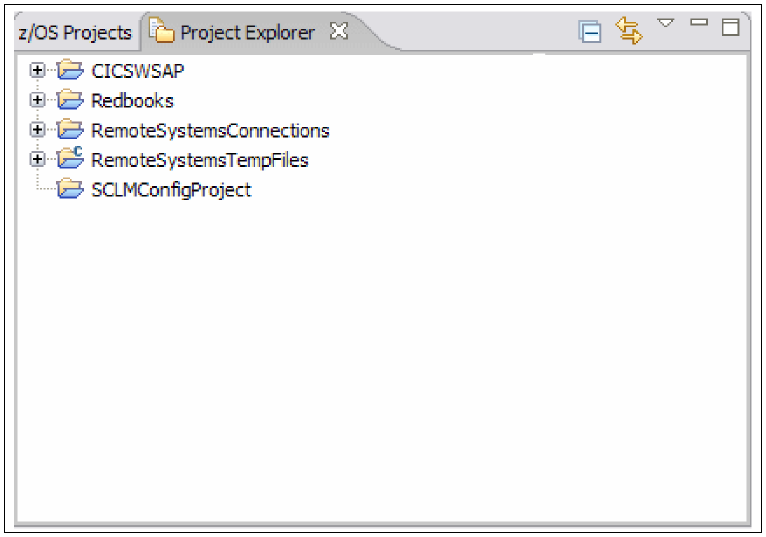
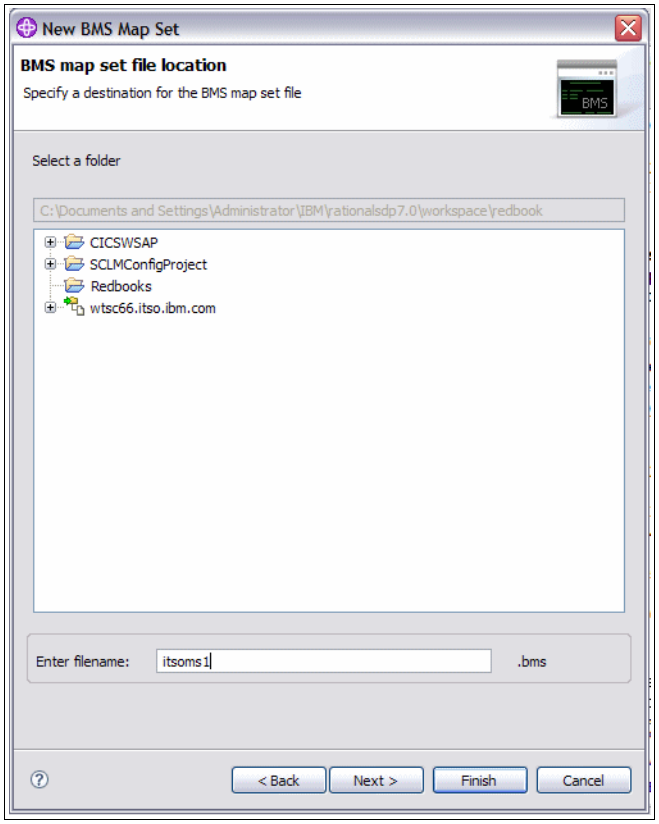


Figure 5-6 Creating the Redbooks local project

5.2.2 Create a new map set

The following steps show you how to create a map set for this project:

1. Click **File** → **New** → **Other** → **zOS**.
2. Select **BMS** → **Map set** from the list of wizards.
3. Click **Next**.
4. Select the folder for the destination of the BMS map set file, and enter the map set name. This is seen in Figure 5-7 on page 94.


New BMS Map Set
BMS map set file location
Specify a destination for the BMS map set file
BMS

Select a folder

C:\Documents and Settings\Administrator\IBM\rationalsdp7.0\workspace\redbook

- CICSWSAP
- SCLMConfigProject
- Redbooks
- wtsc66.itso.ibm.com

Enter filename: itsoms1 .bms

< Back
Next >
Finish
Cancel

Figure 5-7 Defining the Map set name

The map set **itsoms1** are now listed in the Redbooks project in the Project Explorer view, as seen in Figure 5-8 on page 95.

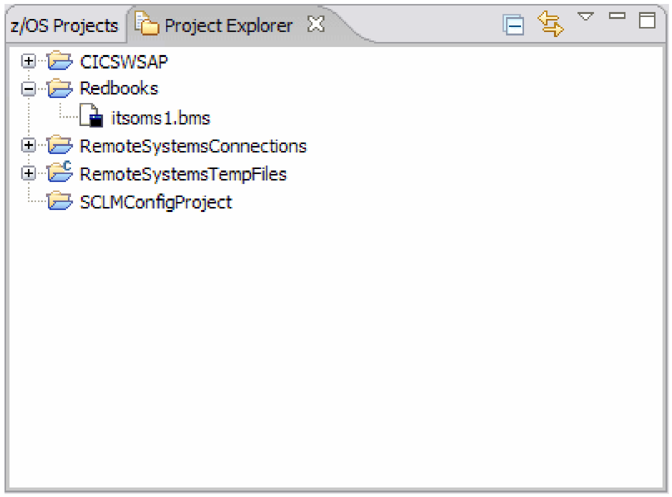


Figure 5-8 The Map set is created

Tip: As a first time user of an Eclipse-based product, if you close views or panes within the current perspective and need to retrieve them again, use **Window** → **Reset Perspective** to return the current perspective to it's default.

5.2.3 Designing the BMS map

Now we use the *drag and drop* technique, dragging from the Palette view and dropping onto our design canvas to create a BMS map.

1. Double-click the map set name in the Project Explorer view to display a blank canvas.
2. Left-click and drag the fields from the Palette that you need, and drop them onto the canvas.
3. Figure 5-9 on page 96 shows that we dropped titles, label, and instruction constant fields and seven input fields into approximate position.

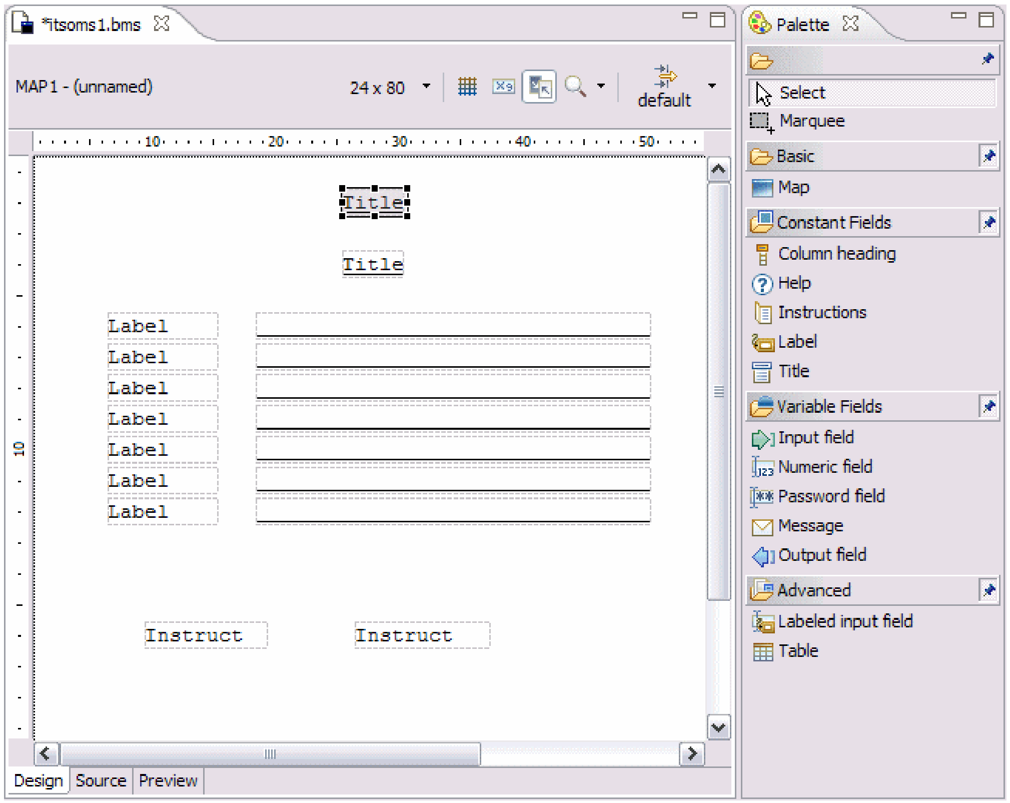


Figure 5-9 The design canvas with Palette items

The Outline view in Figure 5-10 on page 97 shows a list of all the fields associated with the map.

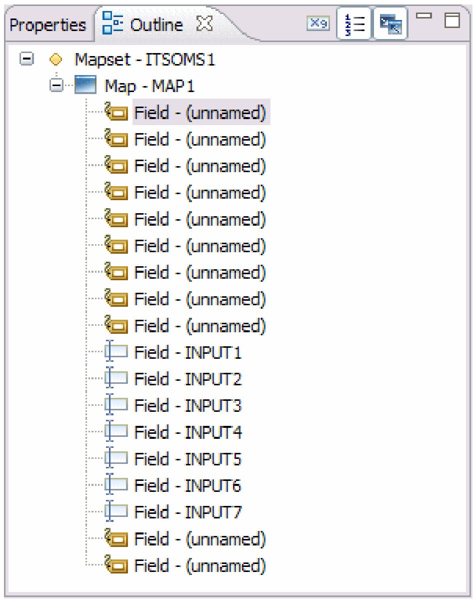

Properties
Outline
Mapset - ITSOMS1
Map - MAP1
Field - (unnamed)
Field - (unnamed)
Field - (unnamed)
Field - (unnamed)
Field - (unnamed)
Field - (unnamed)
Field - (unnamed)
Field - (unnamed)
Field - (unnamed)
Field - INPUT1
Field - INPUT2
Field - INPUT3
Field - INPUT4
Field - INPUT5
Field - INPUT6
Field - INPUT7
Field - (unnamed)
Field - (unnamed)

Figure 5-10 The Outline view shows attribute names for the map

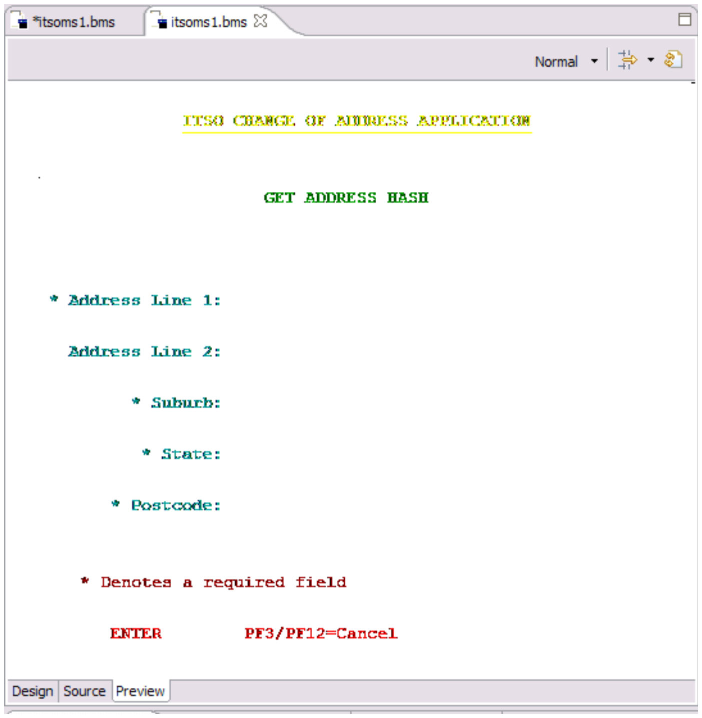
Properties for each of the fields can now be defined by either of the two following methods:

- Double-clicking the field name in the Outline view.
- Right-clicking the field on the design canvas, and selecting **Field Properties**.

The position of the field can be moved by dragging the field around the design canvas.

Tip: When defining the properties of each field, the name of the field must be unique to prevent an error at compile time. The BMS Editor does not syntax check the generated source for duplicate names.

4. View the 3270 representation of the map by clicking the Preview tab, and view or edit the source (if required) by clicking the Source tab. These views are shown in Figure 5-11 on page 98 and in Figure 5-12 on page 99.



ITSO CHANGE OF ADDRESS APPLICATION

GET ADDRESS HASH

* Address Line 1:

Address Line 2:

* Suburb:

* State:

* Postcode:

* Denotes a required field

ENTER PF3/PF12=Cancel

Figure 5-11 Previewing the Map

5. Save the work using the standard eclipse key stroke **Ctrl-S**.

Tip: Some times it appears that the work was not saved, or that some of the panes or views on the workbench go blank. We found that if you exit out of the current map in the design canvas and then double-click the map again in the Projects Explorer View, that the panes become visible again.

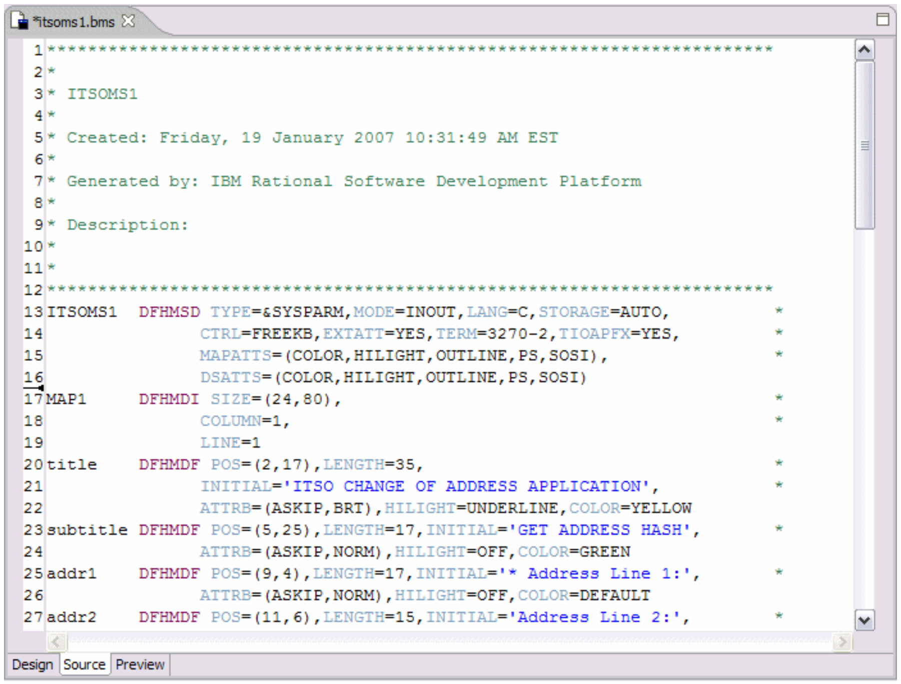

```
***********************************************************************
*
* ITSOMS1
*
* Created: Friday, 19 January 2007 10:31:49 AM EST
*
* Generated by: IBM Rational Software Development Platform
*
* Description:
*
*
***********************************************************************
ITSOMS1  DFHMSD TYPE=&SYSPARM,MODE=INOUT,LANG=C,STORAGE=AUTO,          *
               CTRL=FREEKB,EXTATT=YES,TERM=3270-2,TIOAPFX=YES,         *
               MAPATTS=(COLOR,HILIGHT,OUTLINE,PS,SOSI),                *
               DSATTS=(COLOR,HILIGHT,OUTLINE,PS,SOSI)
MAP1     DFHMDI SIZE=(24,80),                                          *
               COLUMN=1,                                               *
               LINE=1
title    DFHMDF POS=(2,17),LENGTH=35,                                  *
               INITIAL='ITSO CHANGE OF ADDRESS APPLICATION',           *
               ATTRB=(ASKIP,BRT),HILIGHT=UNDERLINE,COLOR=YELLOW
subtitle DFHMDF POS=(5,25),LENGTH=17,INITIAL='GET ADDRESS HASH',       *
               ATTRB=(ASKIP,NORM),HILIGHT=OFF,COLOR=GREEN
addr1    DFHMDF POS=(9,4),LENGTH=17,INITIAL='* Address Line 1:',       *
               ATTRB=(ASKIP,NORM),HILIGHT=OFF,COLOR=DEFAULT
addr2    DFHMDF POS=(11,6),LENGTH=15,INITIAL='Address Line 2:',        *
```

Figure 5-12 The generated map source from the BMS Editor

5.2.4 Creating additional maps

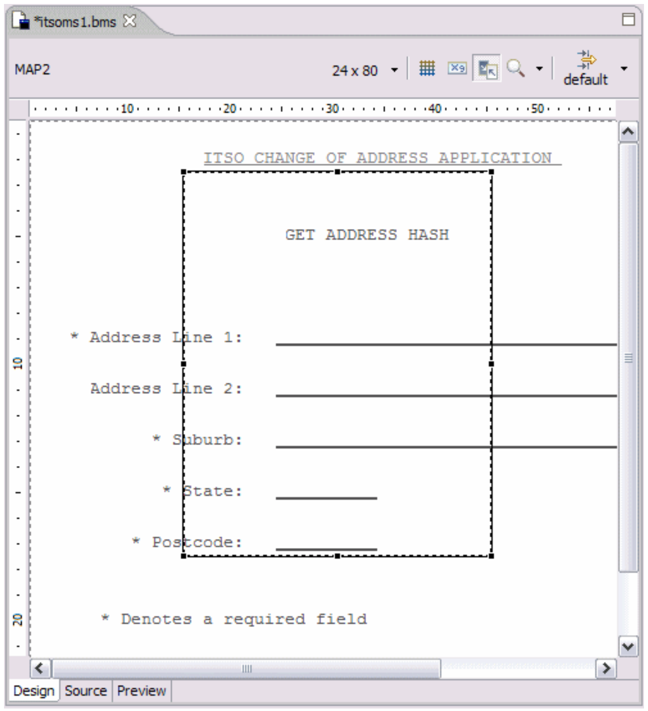
The previous section described the creation of a single map into a map set. Ordinarily there is more than one map within a map set. We will now see how the BMS Editor handles multiple maps within the map set. There are a couple of methods for using multiple maps on the design canvas.

The first and easiest way to manage maps on the canvas is to drag and place the map to the side of the canvas:

1. Click and hold the left mouse button anywhere in the design canvas.
2. Drag and drop the full-window map towards the edge of the canvas view. Note that the size and the position of the map are displayed in the pop-up box.
3. In the Palette, click **Map**.
4. Move the cursor to the design canvas and drop the map.

Alternatively, the map can be ‘hidden’ from view. The Palette contains a Map object that you can drag onto the design canvas to create additional maps. In order to view only the current map, the **Hide Map** option is used to assist with the layering of maps on the work space.

To demonstrate this, Figure 5-13 shows the result of dragging a map object from the Palette onto the design canvas for MAP1, which we created in section 5.2.3, “Designing the BMS map” on page 95, and will be given a default name of MAP2.



```
                ITSO CHANGE OF ADDRESS APPLICATION

                        GET ADDRESS HASH


  * Address Line 1:
    Address Line 2:
          * Suburb:
           * State:
        * Postcode:

      * Denotes a required field
```

Figure 5-13   The map object

The Outline Explorer view shows the newly created MAP2. To work with this map, you may want to hide MAP1 using the Hide Map option on the MAP1 object in the Outline Explorer view. Right-click the MAP1 object as shown in Figure 5-14 on page 101.

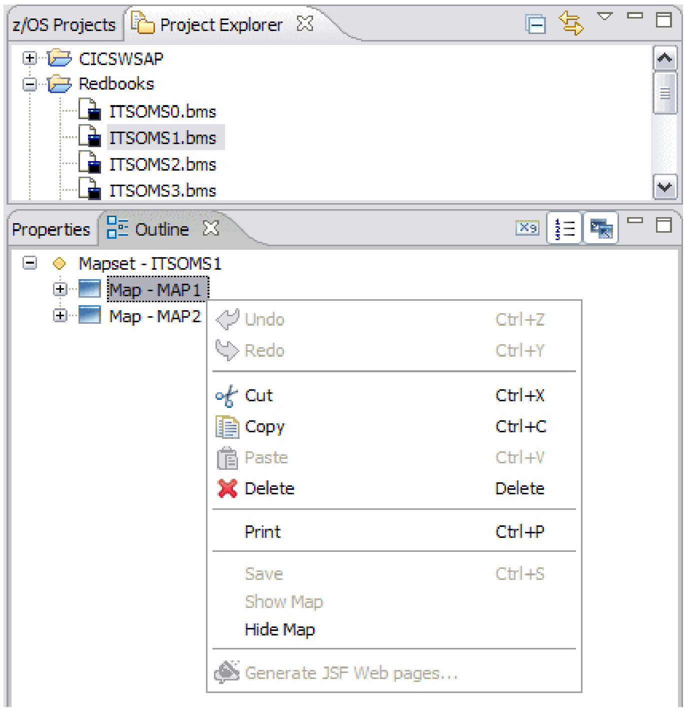


Figure 5-14 The Hide Map Option

Now the design canvas shows only the new map, which you can resize and work on it. The Outline Explorer view shows that the map is hidden as in Figure 5-15 on page 102.

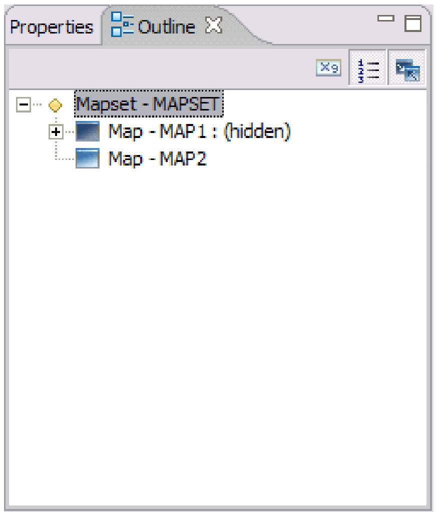


Figure 5-15 The Outline Explorer view showing hidden map

Copying map definitions

Many map sets contain maps that have a common layout or duplicate fields. From the Outline view of the map sets and maps, either the entire map or any of the fields within the map can be copied to another map in either the same or a different map set. Following is the procedure:

1. Right-click the map or field in the Outline view. Multiple fields can be selected by holding the **Ctrl** key and clicking on additional fields.
2. Choose **Copy**.
3. Double-click the destination map set in the Projects view.
4. Click the Outline tab to display the maps in the map set.
5. Right-click the destination map.
6. Choose **Paste**.

5.3 Creating the BMS map set JCL

Traditionally, JCL is used to compile and install the physical and symbolic description maps for use within CICS transactions. Another useful feature of the BMS Editor is functionality to generate JCL for a BMS map set that was created and saved into an existing z/OS project. The JCL is generated and written to an existing PDS on the z/OS.

Important: Ensure that a PDS on the host exists to hold the generated JCL.

We used the following process to generate the JCL for one of our map sets.

5.3.1 Establish a connection to the host

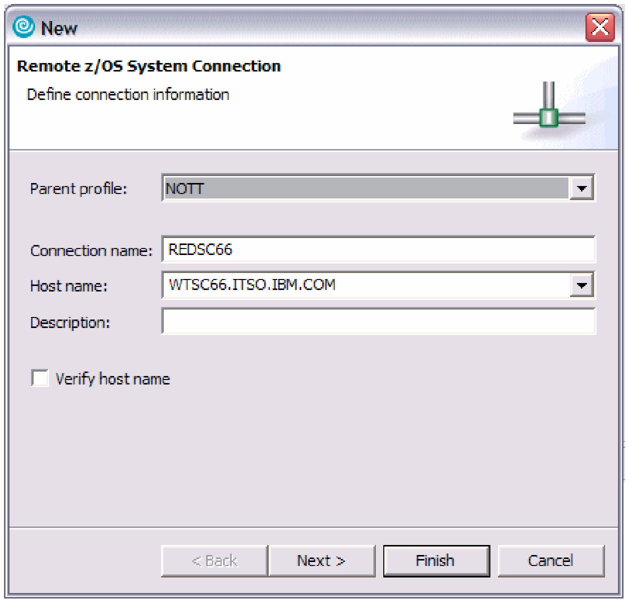
A z/OS project must be created, which is a PC-based representation of artifacts, PDSs, and data set members that also exist on the host. Any objects that are created or edited in the z/OS project are also created on the MVS™ host in real time. This makes development of projects for z/OS almost seamless as there is no need to worry about uploading or creating objects on the host as WD/z does this for you.

Before a z/OS project can be created, a connection to the host system needs to be established.

1. In the Remote Systems view, expand **New Connection** → **z/OS**.

   A window appears, where the connection information is entered. This is shown in Figure 5-16. We called our Connection ***REDSC66***. The host name is ***wtsc66.itso.ibm.com***.

2. Click **Next**. We kept all the defaults.



Parent profile: NOTT

Connection name: REDSC66

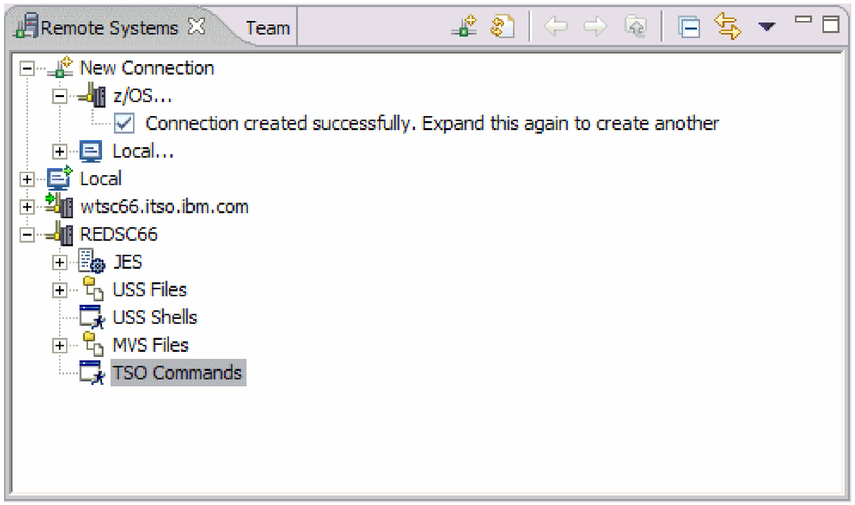
Host name: WTSC66.ITSO.IBM.COM

Description:

☐ Verify host name

Figure 5-16 Defining a Remote Connection

3. Click **Finish**. If the create of the Connection is successful, the Remote Systems view will report this, and there is an explorer view of the host that you can expand to show the file repositories that you want to view in the z/OS environment. This is seen in Figure 5-17.



New Connection
z/OS...
Connection created successfully. Expand this again to create another
Local...
Local
wtsc66.itso.ibm.com
REDSC66
JES
USS Files
USS Shells
MVS Files
TSO Commands

Figure 5-17 A successful connection and Explorer view of the host file groups

5.3.2 Filtering and data set mapping tasks

Now that there is a connection to a host z/OS system, there are a couple of set up tasks to complete. These tasks ensure that the workstation view of the host and the file transfer to and from the host are customized to accommodate local naming conventions and workstation file extensions.

We customize the Remote Systems view of the host file subsystems through the filtering function in WD/z, and ensure that there are mapping rules in place to accommodate the differences in the naming of PDS members and Windows-based filenames.

Filtering the Explorer View of the host connection

The z/OS file systems contain hundreds of data sets and even more members, which you do not necessarily need to display. The tooling allows you to filter the data set and job names, just as you would by using the asterisk on the PDF Dslist function or the prefix command in SDSF on the host system.

As an example, in this book we are interested in the data sets associated with our CICS application **CICSWSAP** and in the jobs associated with our CICS region **SCSCPJA6**. The method to filter just these files is as follows:

1. If not already connected, right-click the connection name in the Remote Systems Explorer view.
2. Click **Connect.**
3. To filter the JES job output to include the jobs associated with our CICS region, right-click **JES** → **New** → **New JES Job Filter**, as seen in Figure 5-18.

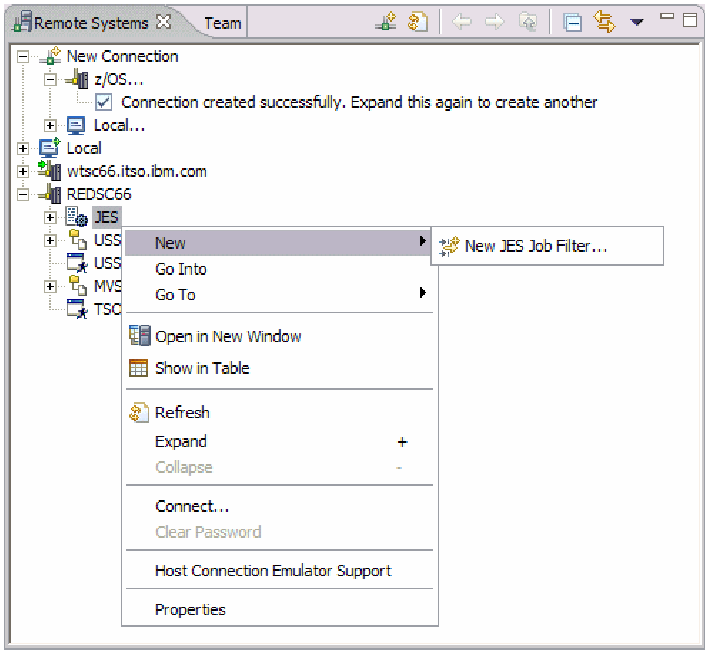
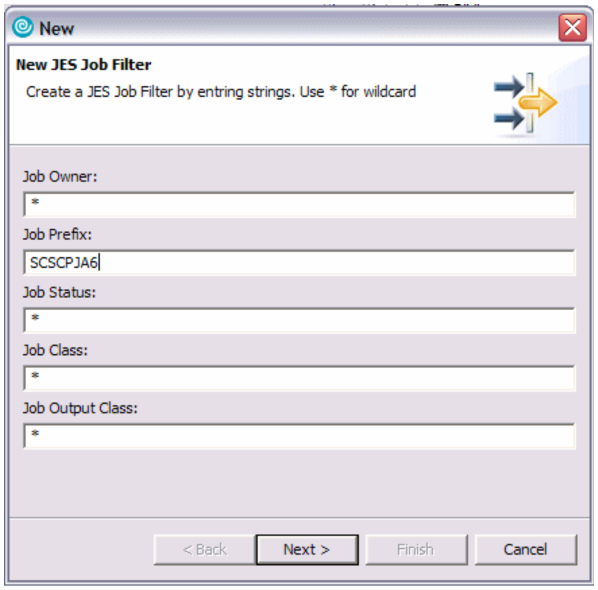


Figure 5-18 Creating a filter for JES jobs

4. Create the filter by entering strings and * for wildcards, as shown in Figure 5-19 on page 106.



Create a JES Job Filter by entring strings. Use * for wildcard

Job Owner: *

Job Prefix: SCSCPJA6

Job Status: *

Job Class: *

Job Output Class: *

Figure 5-19 JES Job Filter

5. Enter a filter name, which will appear in the Explorer tree under the JES category, and then click **Finish**. The expanded view of your selected job has the same effect of sub-command **'?'** in SDSF option ST, and you can view each of the outputs and perform the same sorts of functions as you would with the SDSF sub-commands.
6. Similarly, for data set names, right-click **MVS Files** → **New** → **Filter**, and enter a filter string for the data sets that you want to be displayed in the expanded view of this Explorer tree. The result is shown in Figure 5-20 on page 107.

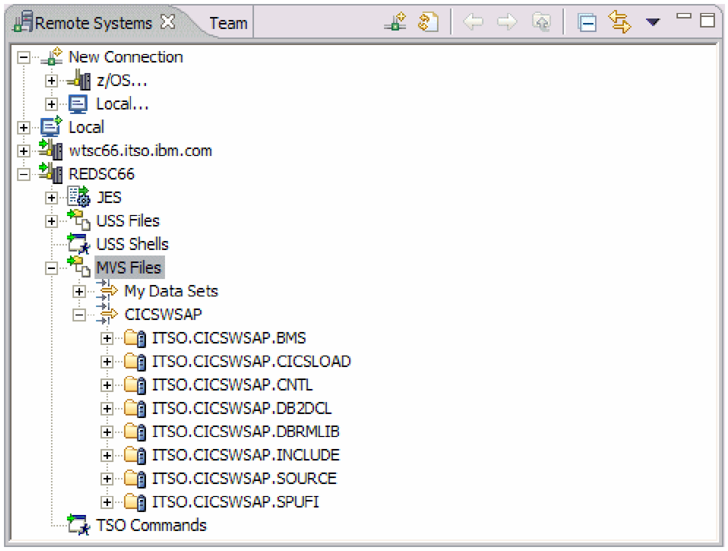


Figure 5-20 A filtered data set view for our project

z/OS File System Mapping

As part of the default z/OS perspective, there is a view on the work space called z/OS File System Mapping that provides a list of mapping criteria for the lowest level qualifier of a PDS and the workstation file extension. If there is a mapping defined, you can see the workstation file extensions in the Remote Systems view of the members in a PDS.

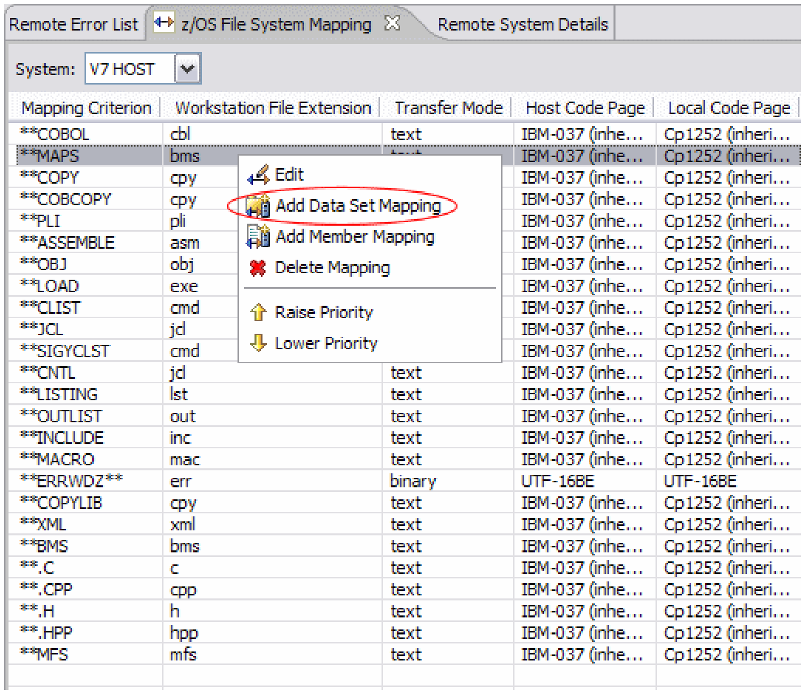
Use the following steps to add a map definition:

1. Open the **z/OS File System Mapping** view.

   Use **Window** → **Show View** → **z/OS File System Mapping** if the view is not already on the work space.

2. Choose the host name in the System pull-down field.

3. Right-click the view, and select **Add Data Set Mapping** as seen in Figure 5-21 on page 108.

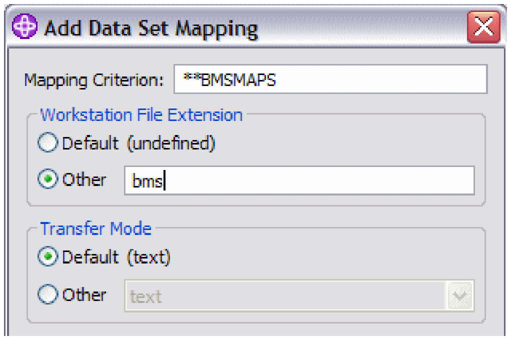


System: V7 HOST

Mapping Criterion	Workstation File Extension	Transfer Mode	Host Code Page	Local Code Page
**COBOL	cbl	text	IBM-037 (inhe...	Cp1252 (inheri...
**MAPS	bms		IBM-037 (inhe...	Cp1252 (inheri...
**COPY	cpy		IBM-037 (inhe...	Cp1252 (inheri...
**COBCOPY	cpy		IBM-037 (inhe...	Cp1252 (inheri...
**PLI	pli		IBM-037 (inhe...	Cp1252 (inheri...
**ASSEMBLE	asm		IBM-037 (inhe...	Cp1252 (inheri...
**OBJ	obj		IBM-037 (inhe...	Cp1252 (inheri...
**LOAD	exe		IBM-037 (inhe...	Cp1252 (inheri...
**CLIST	cmd		IBM-037 (inhe...	Cp1252 (inheri...
**JCL	jcl		IBM-037 (inhe...	Cp1252 (inheri...
**SIGYCLST	cmd		IBM-037 (inhe...	Cp1252 (inheri...
**CNTL	jcl	text	IBM-037 (inhe...	Cp1252 (inheri...
**LISTING	lst	text	IBM-037 (inhe...	Cp1252 (inheri...
**OUTLIST	out	text	IBM-037 (inhe...	Cp1252 (inheri...
**INCLUDE	inc	text	IBM-037 (inhe...	Cp1252 (inheri...
**MACRO	mac	text	IBM-037 (inhe...	Cp1252 (inheri...
ERRWDZ	err	binary	UTF-16BE	UTF-16BE
**COPYLIB	cpy	text	IBM-037 (inhe...	Cp1252 (inheri...
**XML	xml	text	IBM-037 (inhe...	Cp1252 (inheri...
**BMS	bms	text	IBM-037 (inhe...	Cp1252 (inheri...
**.C	c	text	IBM-037 (inhe...	Cp1252 (inheri...
**.CPP	cpp	text	IBM-037 (inhe...	Cp1252 (inheri...
**.H	h	text	IBM-037 (inhe...	Cp1252 (inheri...
**.HPP	hpp	text	IBM-037 (inhe...	Cp1252 (inheri...
**MFS	mfs	text	IBM-037 (inhe...	Cp1252 (inheri...

Figure 5-21 Adding a mapping rule between a PDS member and workstation filename

4. Enter the low-level qualifier of the PDS and the workstation file extension that it is to be mapped to, as in Figure 5-22. In this example we want to map any members in PDS called *.*.BMSMAPS to be given a file extension of .bms on the workstation view.



Mapping Criterion: **BMSMAPS

Workstation File Extension
Default (undefined) [✓ Other] bms

Transfer Mode
[✓ Default (text)] Other text

Figure 5-22 Specifying the mapping rule

5. Click **OK**, and the mapping criterion should now appear in the view.

5.3.3 Create a z/OS Project for the map set

In section 5.2.1, “Create a Project” on page 92 we created a general project that works with objects on the local workstation. To generate JCL for our BMS and work with a host z/OS system, including the utilization of the Editors in WD/z to edit existing BMS map sets, JCL, and other files, we need to create a z/OS Project and associate the map sets to that project.

Use the following steps to create an z/OS Project.

1. Create a connection to the host, as in the previous section.
2. Select **File** → **New** → **Project**.
3. In the New Project panel, expand the z/OS view, and choose **z/OS Project** as in Figure 5-23.

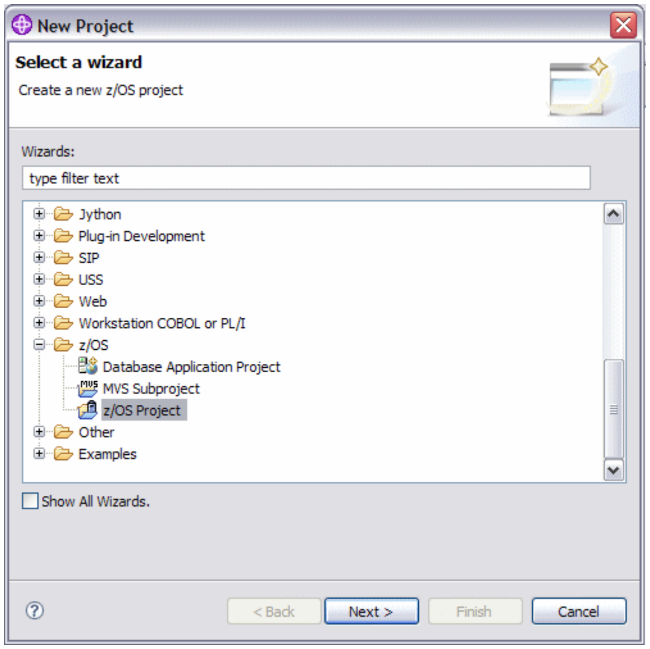
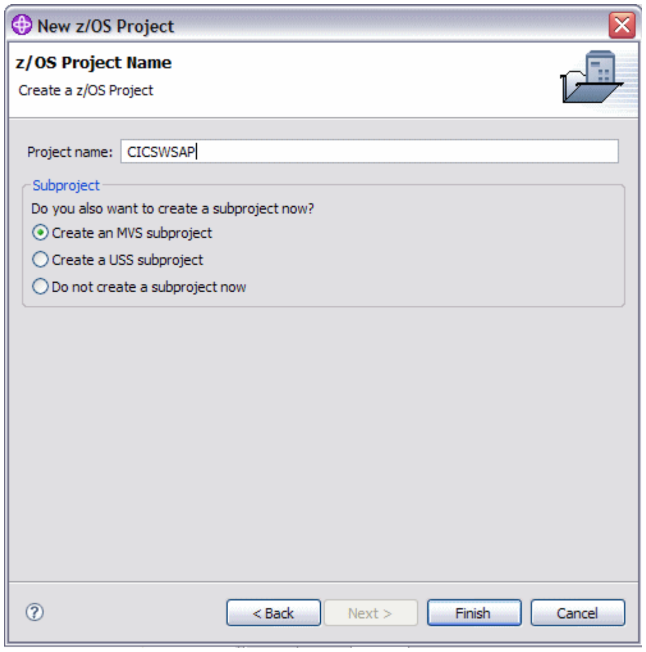


Figure 5-23 Creating a z/OS Project

4. Click **Next** and enter a project name.
5. Check the box to also create an MVS subproject as in Figure 5-24 on page 110.

Note: In WD/z V7, you need to create a subproject for the z/OS project to enable some of the functions to operate on the members of the project.



Project name: CICSWSAP

Subproject

Do you also want to create a subproject now?

(●) Create an MVS subproject

( ) Create a USS subproject

( ) Do not create a subproject now

Figure 5-24 Creating a subproject

6. Click **Finish**, and enter a subproject name.
7. Click **Next**, and take all of the defaults, for now.

   The new project should appear in the z/OS Projects view with the associated subproject and host name. This is seen in Figure 5-25.

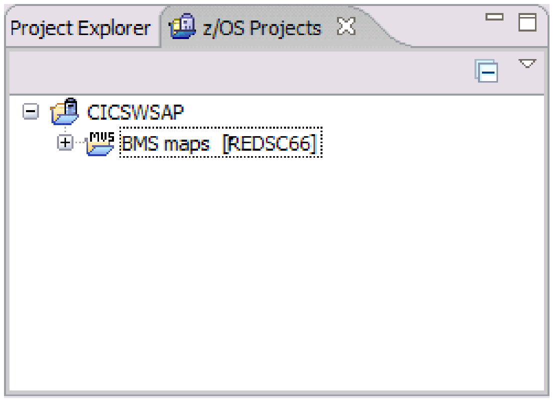


Figure 5-25 z/OS Project Explorer view of z/OS project and subproject

5.3.4 Import map sets into the z/OS Project

You may already have several map sets created in your traditional CICS environment that you want to edit or copy. The BMS Editor provides a modern tooling technique to simplify this process. The BMS Editor can be used on existing map sets on a host system by importing them into a z/OS subproject, or map sets created locally can be copied into the z/OS Project from a local, general project type on the work space.

The effect of copying artifacts from a local project into a z/OS Project is that these are also created in the PDS on the host. In our demonstration, we will now allocate a PDS for the z/OS Project to hold our maps and map sets.

1. Right-click the z/OS subproject name in the z/OS Projects Explorer view.
2. Select **New** → **Allocate PDS**.

   This is shown in Figure 5-26.

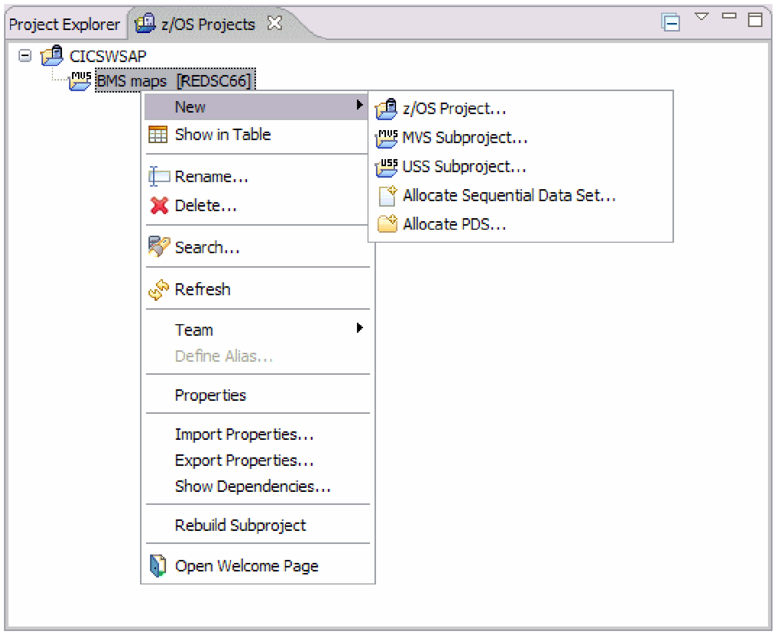


Figure 5-26 Allocating a PDS on the z/OS host

3. Provide a name for the PDS that conforms to your local naming conventions. The BMS Editor in WD/z only recognizes members of the PDS with a file extension of .bms, so check that there is a z/OS File System Mapping rule in effect for the name you give to the PDS. Alternately, you can create a

mapping rule, which is described in “z/OS File System Mapping” on page 107.

Restriction: If you want to use the BMS Editor on existing maps, the PDS created or copied from the host system should have a mapping rule in effect for the file extension .bms. Otherwise only the source can be viewed and edited. There is no design or preview functionality.

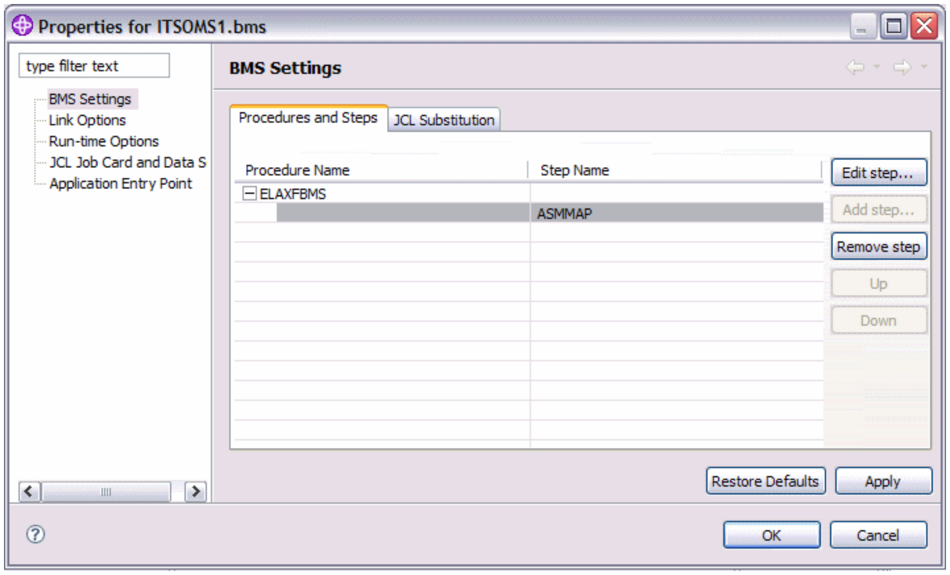
4. Fill in the characteristics of the data set, or take the defaults, and then select **Finish**.

   If filtered, the name of the new PDS is displayed in the Remote Systems view, and it also appears in the z/OS Projects view in the sub-project.
5. Now you can copy and paste map sets from local projects or from host PDSs into this new PDS, or vice versa.
6. Open the maps for edit with the BMS Editor by double-clicking the map.

5.3.5 Create the JCL for the map set

Use the following procedure to create the JCL for assembly of a map set:

1. Right-click the .bms map set file in the z/OS Project folder.
2. Select **Properties**.
3. In the JCL Job Card and Data Set dialog, specify an existing data set to contain the generated JCL.
4. In the BMS Settings dialog, click the step name, and then choose **Edit step** to customize the step name for your site. This is shown in Figure 5-27 on page 113.



Procedure Name	Step Name
ELAXFBMS	
	ASMMAP

Figure 5-27 Customizing map set properties

This brings up an Options view where the destination libraries and procedure name can be edited according to your site specifications.

5. Click **OK** when finished.
6. Right-click the .bms set file name again.
7. Select **Generate JCL** → **For Assemble** or **Generate JCL** → **For Assemble Link**.
8. In the input dialog, specify the target JCL data set name and the member name. You can also choose to set a different job name.
9. Click **OK** to complete the generation process.

Note: If the member name already exists, you are prompted to overwrite the member. If the target PDS does not exists, an error message is displayed.

5.3.6 Submit the JCL and test in CICS

Now we are ready to submit the JCL using the following steps.

1. Right-click the generated JCL member in the data set in the Remote Systems view.
2. Select **Submit**.

3. Use JES to view the output of the job.
4. You can find the output in the library that was specified in the BMS Settings dialog.
5. This map can now be tested in the CICS region with a **`CECI Send Map`** command.

6

Exposing our application as a Web service

In this chapter we demonstrate the steps we took to expose our CICSWSAP application as a Web service. There is some set up in the CICS region to enable Web services support, and we show a few ways that the Web service files, wsdl, and bindings are generated. We discuss the CICS Web Services Assistant, program DFHLS2WS and also show how WebSphere Developer for System z can be used to generate these files. We then test the Web service using the Web Services Explorer in WD/z.

6.1 Configuration for Web service enablement

This section outlines the steps to enable the CICS system for running Web service support. In a basic scenario, HTTP protocol is used as the transport mechanism, although Web service transports are not limited to HTTP. Other transports such as WMQ or JMS could equally be used as well.

The following steps outline how we set up our CICS system to expose our sample application as a Web service provider.

6.1.1 Creating the HFS directories

Web service support requires two HFS directories to be created for our application:

1. A **pickup** directory

   This contains the Web service wsdl and binding files associated with a Web service. The files are suffixed by .wsdl and .wsbind. The directory is specified as the output location for the CICS Web Services Assistant and is associated with a PIPELINE resource definition.

   When a PIPELINE is installed or a CEMT PERFORM SCAN is done on the PIPELINE, this directory is scanned and information in the bindings file is used to dynamically create the WEBSERVICE and URIMAP definitions associated with the PIPELINE. As part of this process, CICS copies any .wsbind files from this directory into a folder in the shelf directory, which is given a name by CICS, unique to the region.

2. A **shelf** directory

   The shelf directory stores the Web service bindings files that are associated with the CICS WEBSERVICE resources. Each of these WEBSERVICE resources is associated with a CICS PIPELINE and this directory is managed by the PIPELINE. Several PIPELINES can use the same shelf directory. These resources are created in section 6.1.2, “Creating the CICS Resources” on page 117.

The directories created for our CICSWSAP application are as follows:

```
/u/jnott/cicswsap/shelf
/u/jnott/cicswsap/wsbind/provider
```

Note: The HFS entries are case sensitive and assume a default CICS HFS install root of /usr/lpp/cicsts.

6.1.2 Creating the CICS Resources

Using the steps in this section, create and install the following resources in CICS:

- PIPELINE
- TCPIPSERVICE
- WEBSERVICE
- URIMAP

1. Configuring the PIPELINE resource

The two components of this definition are the PIPELINE resource itself and a configuration file. The file contains details about the message handlers that will act on Web service requests and responses as they pass though the pipeline. This should contain the message handler programs and the SOAP header processing programs that CICS will invoke when it processes the pipeline.

We start by using the CICS-supplied SOAP 1.1 handler to manage the SOAP envelopes of inbound and outbound requests of our Web service sample. CICS provides sample pipeline configuration files that can be used in both the service provider and service requester scenarios.

Note: Although more than one WEBSERVICE can share a PIPELINE, a single PIPELINE cannot be configured to be both a provider and a requester pipeline. A second pipeline must be configured for outbound requests.

In our scenario, we only require one PIPELINE resource for inbound requests, using our CICS Web service modules in a service provider role. We copied the PIPELINE definition for inbound requests from the CICS supplied group DFH$EXWS. We then altered the definition to suit our environment as shown in Example 6-1 on page 118.

Following are the attributes that we changed:

– **Configfile** specifies the HFS path where the message handler programs and header programs are located.

– **Shelf** specifies the HFS directory where CICS will store the installed wsbind files.

– **Wsdir** specifies the HFS pickup directory where the installable wsbind and WSDL files are located.

Example 6-1 Our PIPELINE provider definition

```
OBJECT CHARACTERISTICS                                    CICS RELEASE =
0640
 CEDA  View PIpeline( WSPIPE01 )
  PIpeline       : WSPIPE01
  Group          : PJA6ADTX
  Description    :
  STatus         : Enabled            Enabled | Disabled
  Configfile     : /usr/lpp/cicsts/cicsts31/samples/pipelines/basicsoap11prov
  (Mixed Case)   : ider.xml
                 :
                 :
                 :
  SHelf          : /u/jnott/CICSWSAP/shelf
  (Mixed Case)   :
                 :
                 :
                 :
  Wsdir          : /u/jnott/CICSWSAP/wsbind/provider
  (Mixed Case)   :
```

2. **Creating a TCPIPSERVICE**

   When a client connects to our Web service over an HTTP service, a TCPIPSERVICE resource is needed to receive the inbound HTTP traffic. We created a TCPIPSERVICE definition using CEDA, or you can use the sample definition in the CICS supplied group DFH$EXWS.

   Following are the attributes that we verified or changed from the sample:

   – **GROup** to specify our resource group PJA6ADTX
   – **STatus** should be Open
   – **POrtnumber** to specify an unused port on the CICS system
   – **PROtocol** should be the default HTTP
   – **TRansaction** should be the default CWXN

3. **Creating WEBSERVICE and URIMAP resources**

   Each function that is exposed as a Web service requires a WEBSERVICE resource to map between the incoming XML of the soap body and the COMMAREA interface of the program, and a URIMAP resource that routes the incoming requests to the correct PIPELINE and WEBSERVICE. Although you can use RDO to define and install these resources, you can also have CICS create them dynamically when you install a PIPELINE resource.

We chose CICS to dynamically create these resources, but before we install the PIPELINE, we need to have the wsbind and WSDL files available. The generation of these files are discussed in the next section, and then these resources are dynamically installed in section 6.1.4, “Installing the PIPELINE resource definitions” on page 121.

6.1.3 Generating the WSBind and WSDL files

In our example, we generate the wsbind and WSDL for one of our business logic functions ITSOGH03. This program simply generates an address hash from an address string passed in a commarea and returns the value in the same commarea. The data structure that maps the commarea is called ITSOGHCA. It is a simple function and can be transformed into a Web service.

The Web Services Assistant easily creates the files needed for the Web service runtime. We customized the supplied job DFHLS2WS, which takes a language structure as input and generates a WSDL file and a WSBIND file.

Following are the steps we took:

1. Create an HFS directory in which to store the generated files—this was already done in a previous step and the name is as follows:

   ```
   /u/jnott/cicswsap/wsbind/provider
   ```

2. Submit the DFHLS2WS job. The job that we ran to generate these files is shown in Example 6-2 on page 120.

Example 6-2 The sample job DFHLS2WS

```
//ITSOLSWS JOB  ,LS2WS,CLASS=A,MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
 //*
 //* Generate WSDL from C Commarea structures
 //*
 //JCL      JCLLIB ORDER=(CICST31B.CICS.SDFHINST)
 //*
 //LS2WS    EXEC DFHLS2WS,
 //      TMPFILE='/tmp',PATHPREF='',USSDIR='cicsts31'
 //INPUT.SYSUT1 DD DATA,DLM='@@'
 PDSLIB=//ITSO.CICSWSAP.INCLUDE
 REQMEM=ITSOGHCA
 RESPMEM=ITSOGHCA
 LANG=C
 LOGFILE=/u/jnott/cicswsap/wsbind/provider/GetHash.log
 MAPPING-LEVEL=1.2
 PGMNAME=ITSOGH03
 URI=cicswsap/GetHash
 PGMINT=COMMAREA
 WSBIND=/u/jnott/cicswsap/wsbind/provider/GetHash.wsbind
 WSDL=/u/jnott/cicswsap/wsbind/provider/GetHash.wsdl
 @@
 //
```

The input to the job includes the following:

PDSLIB - the name of the library that contains the high-level language structures that the application uses to describe the Web service request and the Web service response. In our example, as the structure is a C/370™ data structure, this is the PDS containing the INCLUDEs (rather than copy books for COBOL).

PGMNAME - the name of the program containing the business logic.

LANG - the language of the program.

REQMEM and **RESPMEM** - the names of the INCLUDE members for data mapping of the request and response.

PGMINT - the type of program input—the method that CICS uses to pass data to the application (COMMAREA or container).

LOGFILE, WSBIND, and **WSDL** - the full path names of the files to be generated in the HFS.

URI - the relative HFS address that the client uses to access the Web service.

The generated files are written to the pickup directory in the HFS as specified in the DFHLS2WS JCL.

6.1.4 Installing the PIPELINE resource definitions

When the PIPELINE definition is installed, CICS scans the pickup directory for wsbind files. When CICS finds the wsbind file, it dynamically creates and installs a WEBSERVICE resource definition for it. The name of the WEBSERVICE is derived from the name of the wsbind file.

Then, during the installation of the WEBSERVICE resource, CICS dynamically creates and installs a URIMAP definition, based on the URI specified in the input to DFHLS2WS.

1. We installed the PIPELINE definition using CEDA:

   ```
   CEDA INSTALL PIPELINE(WSPIPE01) GROUP(PJA6ADTX)
   ```

2. To check that the install of the WEBSERVICE and URIMAP were successful, display the WEBSERVICE using CEMT:

   ```
   CEMT INQUIRE WEBSERVICE
   ```

   Example 6-3 shows our new WEBSERVICE resource, named to match the wsbind file in the pickup directory—in our case GetHash.

Example 6-3 The GetHash WEBSERVICE resource

```
 INQUIRE WEBSERVICE
 STATUS:  RESULTS - OVERTYPE TO MODIFY
+ Webs(GetHash                          ) Pip(WSPIPE01)
     Ins Uri($207350 ) Pro(ITSOGH03) Com
     Dat(20070201)
```

Similarly, the URIMAP resource can be viewed using the URI token from the WEBSERVICE definition as in Example 6-4:

```
CEMT INQUIRE URIMAP($207350)
```

Example 6-4 The URIMAP definition

```
INQUIRE URIMAP ($207350)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Uri($207350 ) Pip Ena     Http
    Host(*                          )
 Path(/cicswsap/GetHash               )
```

6.1.5 Performing a scan on the PIPELINE

The PERFORM PIPELINE command initiates a scan of the Web service binding directory that is specified in the WSBIND attribute of the PIPELINE definition.

CICS examines the Web service binding files in the directory to determine if they should be installed into the system:

- CICS installs any files it finds that were not installed already.
- If a file was installed already, but the file in the directory is newer than the one currently in use, then the one that is in use is discarded and the newer file is installed in its place.

Following is the command to perform the scan:

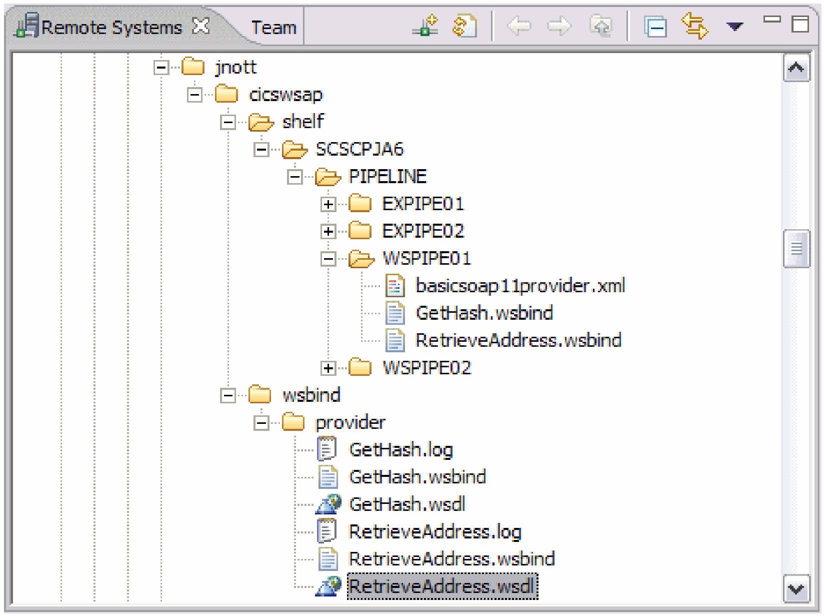
```
CEMT PERFORM PIPELINE(WSPIPE01) SCAN
```

When we created wsbind and WSDL files for further Web services in our example, we used the same PIPELINE (by re-running the DFHLS2WS job). A PIPELINE SCAN was all that was needed to create the WEBSERVICE and URIMAP resources.

Tip: While we were testing, we re-ran the Web services assistant utility DFHLS2WS a few times, which generated newer versions of the wsbind files in our pickup directory. We performed this pipeline scan in order for CICS to pick up the newer wsbind file associated with the pipeline. There are no messages generated on the console, so check MSGUSR for any messages associated with the scan and to see if duplicate entries were found.

6.1.6 Verifying the HFS structure just created

We used WD/z to verify the structures created by CICS and the Web Services Assistant DFHLS2WS. This can also be done via UNIX System Services on the host, although the Remote Systems view gives a clearer picture of the files we created from the previous steps. This is shown in Figure 6-1 on page 123.



```
jnott
  cicswsap
    shelf
      SCSCPJA6
        PIPELINE
          EXPIPE01
          EXPIPE02
          WSPIPE01
            basicsoap11provider.xml
            GetHash.wsbind
            RetrieveAddress.wsbind
          WSPIPE02
    wsbind
      provider
        GetHash.log
        GetHash.wsbind
        GetHash.wsdl
        RetrieveAddress.log
        RetrieveAddress.wsbind
        RetrieveAddress.wsdl
```

Figure 6-1 WD/z Remote System view of our Web service HFS structure

From the view in Figure 6-1, there are a few points to note:

– The DFHLS2WS job generated all the files in the pickup directory:

```
jnott/cicswsap/wsbind/provider
```

– CICS generated all the directories and files in the shelf directory as a result of the **`PERFORM PIPELINE(WSPIPE01) SCAN`** :

```
jnott/cicswsap/shelf/SCSCPJA6/PIPELINE/WSPIPE01
```

6.2 Using WD/z to generate WSDL

As part of the functionality in WD/z V7 for developing z/OS applications and Web services are the Enterprise Service Tools for Web services and SOAP. Within this functionality is the Web services for CICS wizard that provides the functionality similar to that of the DFHLS2WS utility discussed in the previous section.

The example we are going to use is to generate the business logic in program ITSOCA03 using the COBOL copy book called ITSOCACD, which provides the data structure for our business logic for Listing Corporate Acknowledgements.

Unlike our other data structures, this structure represents a CONTAINER rather than a COMMAREA.

The following steps demonstrate the generation of wsdl and wsbind files from our developed COBOL copy book data structure.

6.2.1 Importing the COBOL copy book

Use the following steps to import the copy book member from the z/OS remote system into a local project in the Navigator view:

1. Verify that the Navigator view is open.
2. Use **Window** → **Show View** → **Other**, and expand **General**.
3. Select **Navigator**.
4. If you do not have a local project, create one using **File** → **New** → **Project** and expand **General**.
5. Select **Project**.
6. Drag and drop or import the copybook from the PDS containing the member in the Remote Systems view of the host into the local project.

Tip: The copy book in the local project to be used by the Enterprise Service Tools Wizard must have a file extension of .cbl, .cpy, .cob, or .ccp.

You can check that the file extension was mapped to your PDS low level qualifier in the z/OS File System Mapping view, so that when the file is copied across from PDS to the workstation project, it has the correct file extension. z/OS File System Mapping was discussed in Chapter 5.

6.2.2 Running the Web Services for CICS wizard

1. Right-click the copy book in the Navigator view, and select **Enable Enterprise Web Service.**

   This is shown in Figure 6-2 on page 125.

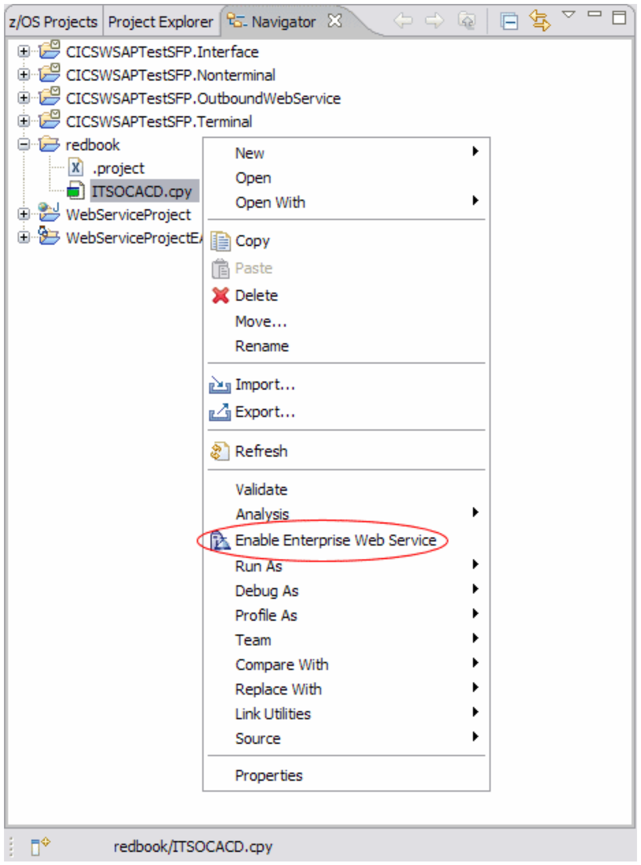


Figure 6-2 Select the copy book to generate Web Service artifacts

2. This starts the Enterprise Service Tools wizard as in Figure 6-3 on page 126. Check that the following are selected:

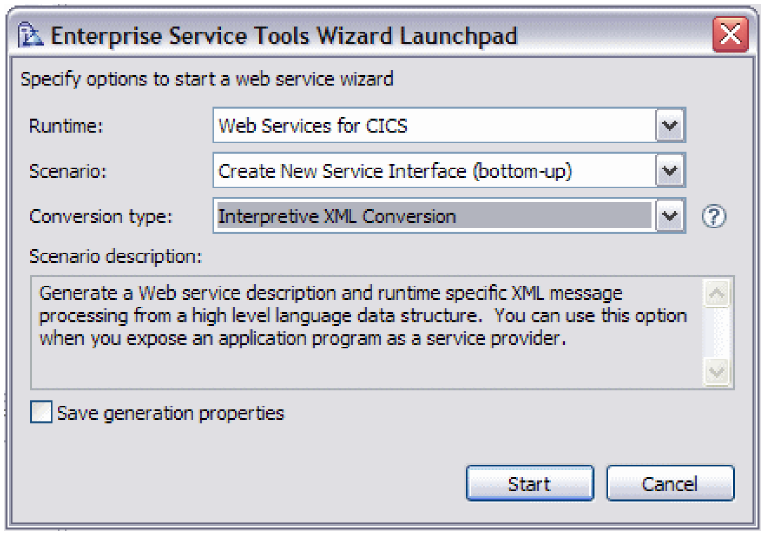
   – **Runtime**: Web Services for CICS

   – **Scenario**: Create New Service Interface (bottom-up)

   – **Conversion type**: Interpretive XML conversion

     Using the bottom-up scenario with Interpretive XML conversion generates files to implement a Web service provider for the Web services CICS environment. Interpretive XML conversion is supported only for the Web services for CICS runtime environment.

3. Click **Start**. This starts up the Web Services for CICS wizard as in Figure 6-4 on page 127. This wizard performs functionality similar to that of the DFHLS2WS utility in the CICS Web Services Assistant.



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

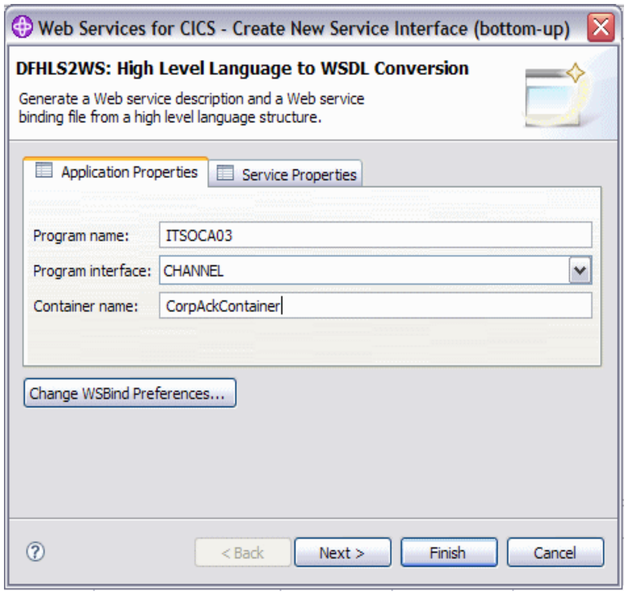
Scenario description:

Generate a Web service description and runtime specific XML message processing from a high level language data structure. You can use this option when you expose an application program as a service provider.

☐ Save generation properties

Figure 6-3 The Enterprise Service Tools Wizard Launchpad

4. On the Application properties tab, specify the following options (Figure 6-4 on page 127):
   - **Program name**: The name of the CICS application program that is to be exposed as a Web service.
   - **Program Interface**: Specifies that CICS passes data to the application program by way of a channel, rather than a COMMAREA.
   - **Container name**: Specifies the name of the container that holds the high level data structure.


Web Services for CICS - Create New Service Interface (bottom-up)

DFHLS2WS: High Level Language to WSDL Conversion

Generate a Web service description and a Web service binding file from a high level language structure.

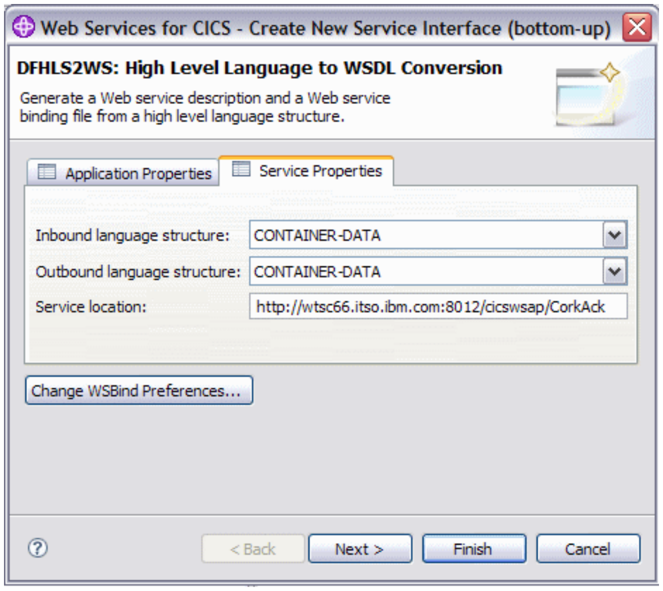
Program name: ITSOCA03

Program interface: CHANNEL

Container name: CorpAckContainer

Figure 6-4 Specify the CONTAINER name

5. On the Service Properties tab specify the following options, as seen in Figure 6-5 on page 128.
   - **Inbound language structure**: Specifies the high-level structure for the Web service request—in this case it is CONTAINER-DATA.
   - **Outbound language structure**: Specifies the high-level structure for the Web service response—in this case it is CONTAINER-DATA.
   - **Service Location**: Specifies the location of the Web service on the host.


Web Services for CICS - Create New Service Interface (bottom-up)

DFHLS2WS: High Level Language to WSDL Conversion

Generate a Web service description and a Web service binding file from a high level language structure.

Inbound language structure: CONTAINER-DATA

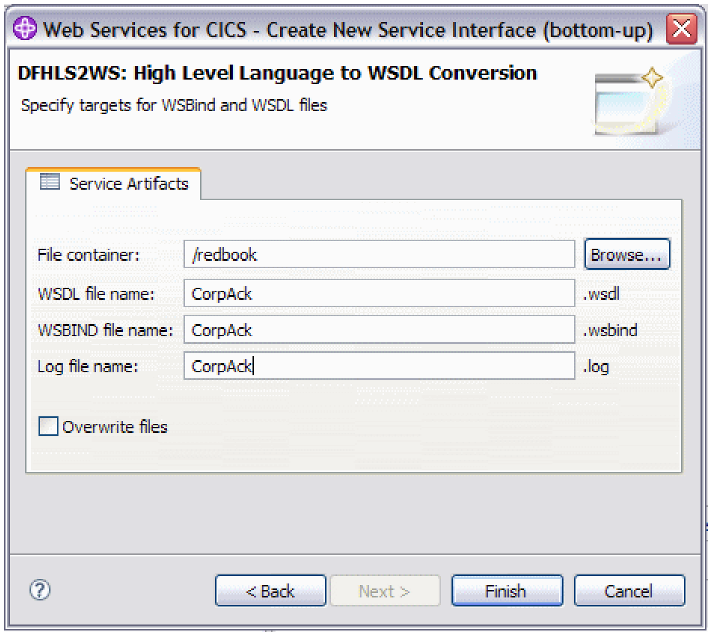
Outbound language structure: CONTAINER-DATA

Service location: http://wtsc66.itso.ibm.com:8012/cicswsap/CorkAck

Figure 6-5 Specify CONTAINER-DATA as the language structure

6. Click **Next**.
7. The panel displayed in Figure 6-6 on page 129 allows you to specify the target names of the generated artifacts. By default the target location is the local file system. You can also specify the remote UNIX System Services file path name if you want to generate these directly into the remote host by browsing the file container for the appropriate file.

   We also specify that the target program communicates via a channel, and we name the expected CONTAINER.
8. Click **Next**.



Web Services for CICS - Create New Service Interface (bottom-up)

DFHLS2WS: High Level Language to WSDL Conversion

Specify targets for WSBind and WSDL files

Service Artifacts

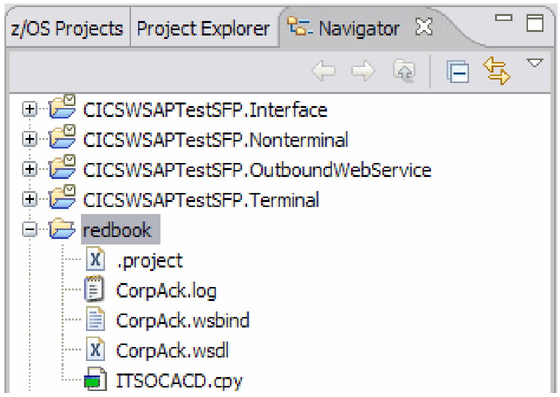
File container:	/redbook	Browse...
WSDL file name:	CorpAck	.wsdl
WSBIND file name:	CorpAck	.wsbind
Log file name:	CorpAck	.log

☐ Overwrite files

Figure 6-6 WSBind and WSDL file target names

9. Click **Finish**.

The generated artifacts can now be seen in the Explorer view of the local project.



```
CICSWSAPTestSFP.Interface
CICSWSAPTestSFP.Nonterminal
CICSWSAPTestSFP.OutboundWebService
CICSWSAPTestSFP.Terminal
redbook
    .project
    CorpAck.log
    CorpAck.wsbind
    CorpAck.wsdl
    ITSOCACD.cpy
```

Figure 6-7 The generated Web service artifacts

6.2.3 Creating the CICS resources

We generated the Web service wsdl and bind files, which brings us to the same point as having run the DFHLS2WS utility in section 6.1.3, “Generating the WSBind and WSDL files” on page 119. In this example we wrote the generated files to a local project on the workstation, so in order for CICS to dynamically create the WEBSERVICE and URIMAP resources, we need to copy the .wsdl and .wsbind files to our pickup directory for our PIPELINE in the UNIX System Services HFS.

To do this we performed a drag and drop of the two files CorpAck.wsbind and CorpAck.wsdl to the HFS directory: `u/jnott/cicswsap/wsbind/provider`.

Now we can scan the pipeline to dynamically create the CICS resources as we did in section 6.1.5, “Performing a scan on the PIPELINE” on page 122, by issuing the following CICS command:

```
CEMT PERFORM PIPELINE(EXPIPE01) SCAN
```

This command detects the new bindings file in the pickup directory, copies it to the shelf directory for this CICS region, and dynamically creates the WEBSERVICE and URIMAP resources.

6.3 Testing the Web service

Now that all of the resources are created in CICS, and the Web service bind files and wsdl are generated, the service is ready to be tested. A simple way to achieve this is by using the Web Services Explorer in WD/z. The only input required is the wsdl file for the service.

From a previous step the wsdl was generated and saved in the HFS on the remote host. The following steps show how we tested our sample GetHash.wsdl Web service.

1. **Import a copy of the wsdl file into a local project on your work space**.

   To do this:

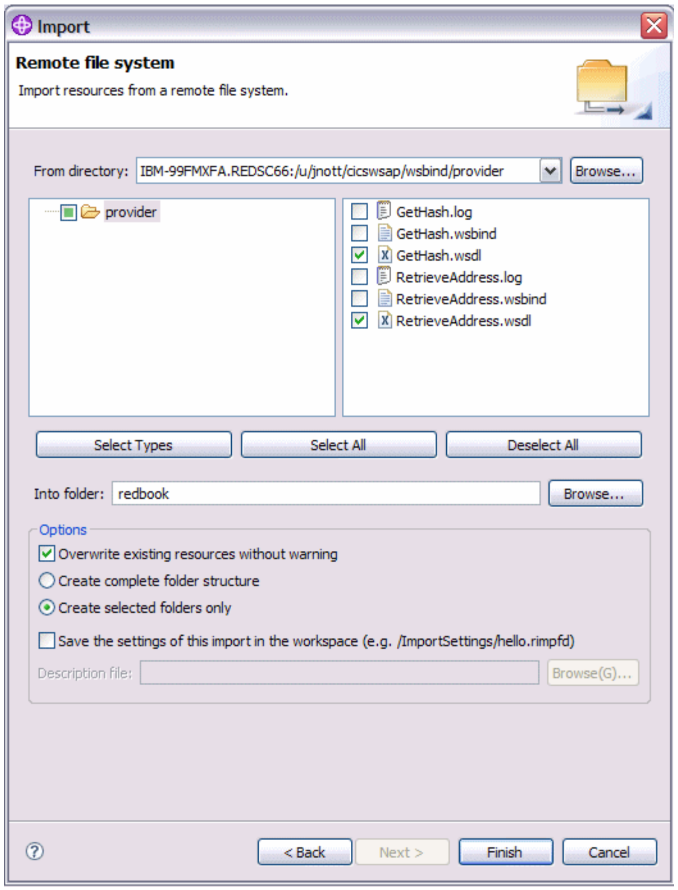
   a. Select **File** → **Import**.

   b. In the Select panel, choose **Other** → **Remote File System**.

   c. Click **Next**.

   d. Browse for the location of the folder containing the wsdl.

   e. In the Remote file system panel, select the wsdl to import. This is shown in Figure 6-8 on page 131.


Import

Remote file system

Import resources from a remote file system.

From directory: IBM-99FMXFA.REDSC66:/u/jnott/cicswsap/wsbind/provider

provider

- [ ] GetHash.log
- [ ] GetHash.wsbind
- [✓] GetHash.wsdl
- [ ] RetrieveAddress.log
- [ ] RetrieveAddress.wsbind
- [✓] RetrieveAddress.wsdl

Into folder: redbook

Options
- [✓] Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only
- [ ] Save the settings of this import in the workspace (e.g. /ImportSettings/hello.rimpfd)

Description file:

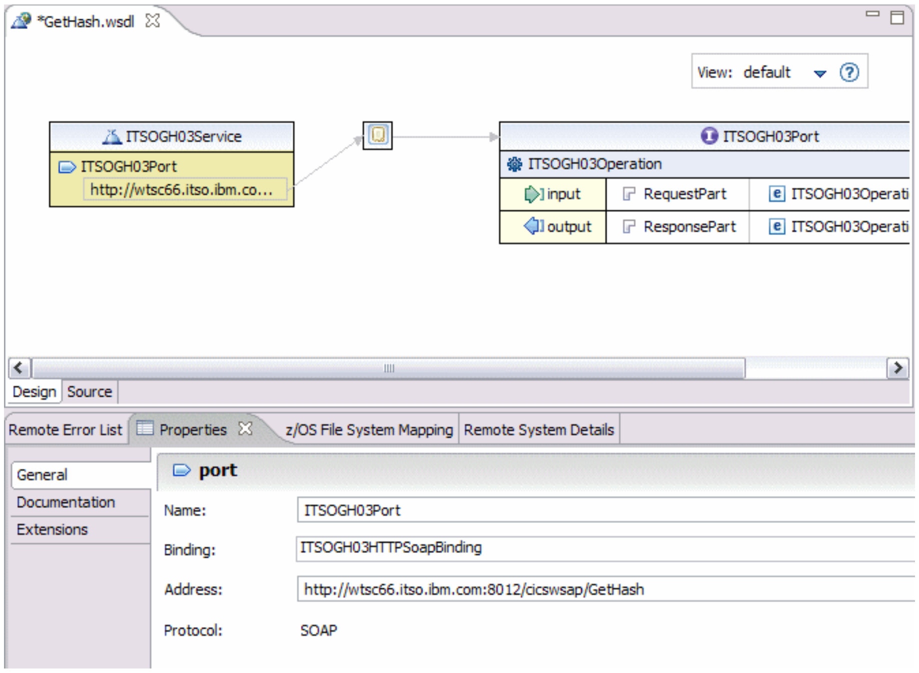
Figure 6-8 Select the remote wsdl to import

   f. Click **Finish**.
   g. The imported files are now displayed in the local project.

2. **Define the location of the Web service on the host.**

   Use the following steps to do this:

   a. Select the GetHash.wsdl in the local project, and double-click to open up the WSDL editor.
   b. Choose the Design view of the wsdl object.
   c. Click the Port for the service.
   d. In the Properties pane, select the **General** tab, and enter the address of the service. This is the URL of the remote host and the URL of the Web service. This is shown in Figure 6-9 on page 132.



Name: ITSOGH03Port
Binding: ITSOGH03HTTPSoapBinding
Address: http://wtsc66.itso.ibm.com:8012/cicswsap/GetHash
Protocol: SOAP

Figure 6-9 Use the WSDL Editor to specify location

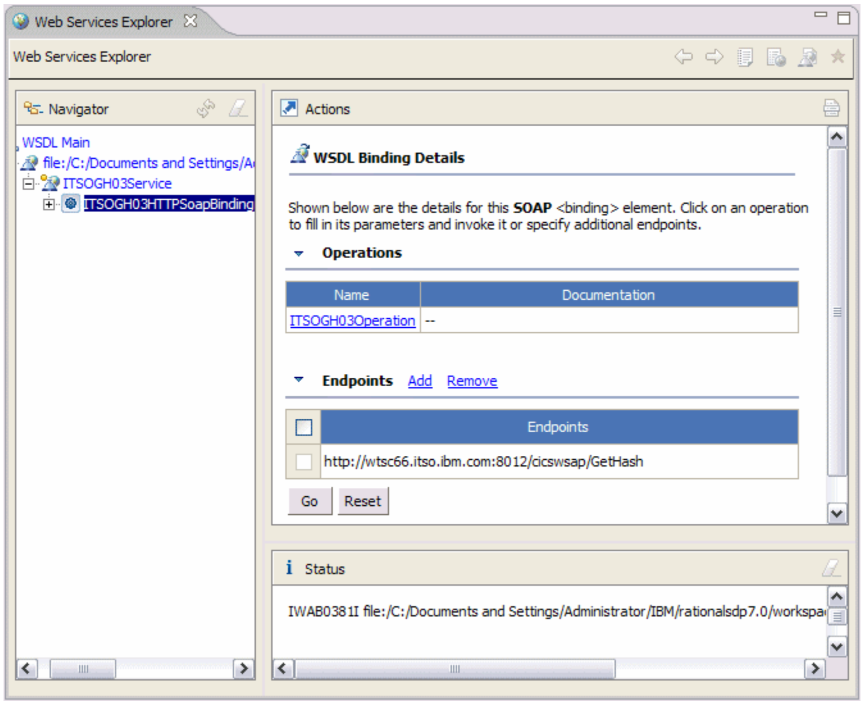
3. **Use the Web Services Explorer to test the Web service.**

   To start the Web Services Explorer, right-click the wsdl file, and select **Web Services** → **Test with Web Services Explorer.** This opens a window that has three panes in it as seen in Figure 6-10 on page 133:

   – The **Navigator** pane shows a history of all previously tested WSDL files and services. Bindings and operations can be viewed.

   – The **Actions** pane is used to execute an operation or to change an endpoint at runtime.

   – The **Status** pane shows any output messages and SOAP envelopes generated for the service.



Shown below are the details for this **SOAP** <binding> element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.

Name	Documentation
ITSOGH03Operation	--

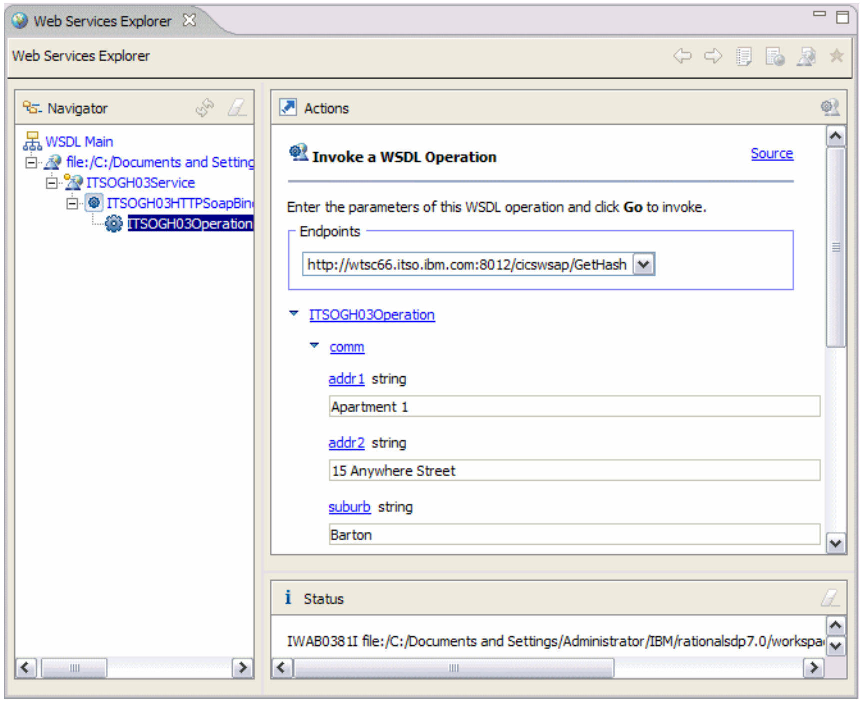
☐	Endpoints
☐	http://wtsc66.itso.ibm.com:8012/cicswsap/GetHash

IWAB0381I file:/C:/Documents and Settings/Administrator/IBM/rationalsdp7.0/workspa

Figure 6-10 The Web Services Explorer in WD/z

To issue the Web service request for testing, click the Operation name (ITSOGH03Operation), and the Action name expands to show the data representation of the service you want to invoke. This is seen in Figure 6-11 on page 134.

The labels correspond to the names of the fields in the commarea. In our example, our logic takes five address fields as input and returns three fields as output. As we are using the same commarea and have not written a wrapper to manipulate the request and response data, all the fields in the commarea are displayed in the request and the response.



Enter the parameters of this WSDL operation and click **Go** to invoke.

Endpoints: http://wtsc66.itso.ibm.com:8012/cicswsap/GetHash

ITSOGH03Operation
comm
addr1 string: Apartment 1
addr2 string: 15 Anywhere Street
suburb string: Barton

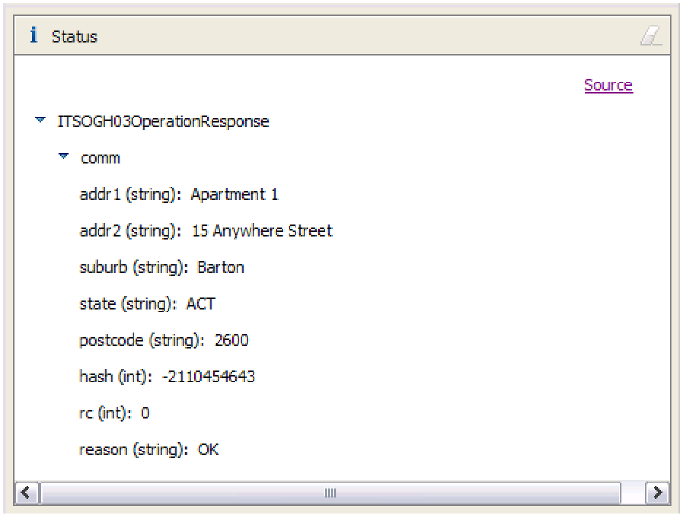
Status

IWAB0381I file:/C:/Documents and Settings/Administrator/IBM/rationalsdp7.0/workspa

Figure 6-11 Enter the request specific data for the Web service on the Action pane

Enter some data to test, and click **Go**.

The result, including any error messages, appear in the Status pane as in Figure 6-12 on page 135, showing the contents of the ITSOGH03OperationResponse.



```
ITSOGH03OperationResponse
  comm
    addr1 (string):  Apartment 1
    addr2 (string):  15 Anywhere Street
    suburb (string):  Barton
    state (string):  ACT
    postcode (string):  2600
    hash (int):  -2110454643
    rc (int):  0
    reason (string):  OK
```

Figure 6-12 The expected results of the Web service test

The test was successful!

7

Configuring publication/subscription

In this chapter we configure the publication/subscription (pub/sub) facility of WebSphere Message Broker V6, which provides an essential component of our *Change of Address* application.

As you may not be familiar with WebSphere Message Broker, we start this chapter with a brief overview of the functions and facilities of WebSphere Message Broker V6. If you are familiar with this topic, you can skip straight to 7.2, “Establishing the pub/sub environment” on page 165.

We chose to run pub/sub on our System z broker. This is not significantly different from implementing the WMB pub/sub on any other platform, in fact, as this chapter shows, we originally implemented the pub/sub message flow on Windows and then ported to the System z broker, which was a trivial exercise.

7.1 Introduction to WebSphere Message Broker

WebSphere Message Broker (WMB) falls into the category of software known as Application Integration or sometimes called *middleware*. Its purpose is to integrate software components that may not otherwise communicate directly. An example might be a proprietary accounting package on one side and a CICS application on the other. WMB allows the reformatting of messages from one system to be used by the other. Business rules can be applied to the data that is flowing through the message broker in order to route, store, retrieve, transform, and enrich the information.

7.1.1 Capabilities of WebSphere Message Broker

The primary capabilities of WebSphere Message Broker are message routing, message transformation, message enrichment, and publish/subscribe. Together these capabilities make WebSphere Message Broker a powerful tool for business integration.

Message routing

WebSphere Message Broker provides connectivity for both standards based and non-standards based applications and services. The routing can be simple point-to-point routing or it can be based on matching the content of the message to business rules defined to the broker.

WebSphere Message Broker contains a choice of transports that enable secure business to be conducted at any time and any place, providing powerful integration using mobile, telemetry, and Internet technologies. 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 supports the following transports:

- WebSphere MQ Enterprise Transport
- WebSphere MQ Web Services Transport
- WebSphere MQ Real-time Transport
- WebSphere MQ Multicast Transport
- WebSphere MQ Mobile Transport
- WebSphere MQ Telemetry Transport
- WebSphere Broker JMS Transport

In addition to the supplied transports, the facility exists for users to write their own input nodes to accept messages from other transports and formats as defined by the user.

Message transformation and enrichment

Transformation and enrichment of in-flight messages is an important capability of WebSphere Message Broker, and it allows for business integration without the need for any additional logic in the applications themselves.

Messages can be transformed between applications to use different formats, for example, transforming from a custom format in a traditional system to XML messages that can be used with a Web service. This provides a powerful mechanism to unify organizations because business information can now be distributed to applications that handle completely different message formats without a need to reprogram or add to the applications themselves.

Messages can also be transformed and enriched by integration with multiple sources of data such as databases, applications, and files. This allows for any type of data manipulation including logging, updating, and merging. For the messages that flow through the broker, business information can be stored in databases or can be extracted from databases and files and added to the message for processing in the target applications.

Complex manipulation of message data can be performed using the facilities provided in the Message Brokers Toolkit, such as ESQL and Java.

In WebSphere Message Broker, message transformation and enrichment depends on a broker understanding the structure and content of the incoming message. Self-defining messages, such as XML messages, contain information about their own structure and format. However, before other messages, such as custom format messages, can be transformed or enhanced, a message definition of their structure must exist. The Message Brokers Toolkit contains facilities for defining messages to the message broker. These facilities are discussed in more detail in the following sections.

Publish/subscribe

The simplest way of routing messages is to use point-to-point messaging to send messages directly from one application to another. Publish/subscribe provides an alternative style of messaging in which messages are sent to all applications that have subscribed to a particular topic.

The broker handles the distribution of messages between publishing applications and subscribing applications. As well as applications publishing on or subscribing to many topics, more sophisticated filtering mechanisms can be applied.

An improved flow of information around the business is achieved through the use of publish/subscribe and the related technology of multicast. These flexible distribution mechanisms move away from hard-coded point-to-point links.

7.1.2 Components of WebSphere Message Broker

WebSphere Message Broker is comprised of the following two principle parts:

- A development environment for the creation of message flows, message sets, and other message flow application resources.
- A runtime environment, which contains the components for running those message flow applications that are created in the development environment.

Development environment

The development environment is where the message flow applications that provide the logic to the broker are developed. The broker uses the logic in the message flow applications to process messages from business applications at run time. In the Message Brokers Toolkit, you can develop both message flows and message sets for message flow applications.

Message flows

Message flows are programs that provide the logic that the broker uses to process messages from business applications. Message flows are created by connecting nodes together, with each node providing some basic logic. A selection of built-in nodes is provided with WebSphere Message Broker for performing particular tasks. These tasks can be combined to perform complex manipulations and transformations of messages.

A choice of methods is available for defining the logic in the message flows to transform data. Depending on the different types of data or the skills of the message flow application developer, the following options are available:

- Extended Structured Query Language (ESQL)
- Java
- eXtensible Style sheet Language for Transformations (XSLT)
- Drag-and-drop mappings

The nodes in the message flows define the source and the target transports of the message, any transformations and manipulations based on the business data, and any interactions with other systems such as databases and files.

Message sets

A message set is a definition of the structure of the messages that are processed by the message flows in the broker. As mentioned previously, in order for a message flow to be able to manipulate or transform a message, the broker must know the structure of the message. The definition of a message can be used to verify message structure and to assist with the construction of message flows and mappings in the Message Brokers Toolkit.

Message sets are compiled for deployment to a broker as a message dictionary, which provides a reference for the message flows to verify the structure of messages as they flow through the broker.

Broker Application Development perspective

The Broker Application Development perspective is the part of the Message Brokers Toolkit that is used to design and develop message flows and message sets. It contains editors that create message flows, create transformation code such as ESQL, and define messages.

Runtime environment

The runtime environment is the set of components that are required to deploy and run message flow applications in the broker.

Broker

The broker is a set of application processes that host and run message flows. When a message arrives at the broker from a business application, the broker processes the message before passing it on to one or more business applications. The broker routes, transforms, and manipulates messages according to the logic that is defined in their message flow applications.

A broker uses WebSphere MQ as the transport mechanism both to communicate with the Configuration Manager from which it receives configuration information and to communicate with any other brokers to which it is associated.

Each broker has a database in which it stores the information that it needs to process messages at run time.

Execution groups

Execution groups enable message flows within the broker to be grouped together. Each broker contains a default execution group. Additional execution groups can be created as long as they are given unique names within the broker. Each execution group is a separate operating system process; therefore, the contents of an execution group remain separate from the contents of other execution groups within the same broker. This can be useful for isolating pieces of information for security because the message flows execute in separate address spaces or as unique processes.

Message flow applications are deployed to a specific execution group. To enhance performance, the same message flows and message sets can be running in different execution groups.

Configuration Manager

The Configuration Manager is the interface between the Message Brokers Toolkit and the brokers in the broker domain. The Configuration Manager stores configuration details for the broker domain in an internal repository, providing a central store for resources in the broker domain.

The Configuration Manager is responsible for deploying message flow applications to the brokers. The Configuration Manager also reports back on the progress of the deployment and on the status of the broker. When the Message Brokers Toolkit connects to the Configuration Manager, the status of the brokers in the domain is derived from the configuration information stored in the Configuration Manager's internal repository.

Broker domain

Brokers are grouped together in broker domains. The brokers in a single broker domain share a common configuration that is defined in the Configuration Manager. A broker domain contains one or more brokers and a single Configuration Manager. It can also contain a User Name Server. The components in a broker domain can exist on multiple machines and operating systems and are connected together with WebSphere MQ channels.

A broker belongs to only one broker domain.

User Name Server

A User Name Server is an optional component that is required only when publish/subscribe message flow applications are running and where extra security is required for applications to be able to publish or subscribe to topics. The User Name Server provides authentication for topic-level security for users and groups that are performing publish/subscribe operations.

Broker Administration perspective

The Broker Administration perspective is the part of the Message Brokers Toolkit that is used for the administration of any broker domains that are defined to the Message Brokers Toolkit. This perspective is also used for the deployment of message flows and message sets to brokers in the defined broker domains.

The Broker Administration perspective also contains tools for creating message broker archive (bar) files. Bar files deploy message flow application resources such as message flows and message sets.

The Broker Administration perspective also contains tools to help test message flows. These tools include Enqueue and Dequeue for putting and getting messages from WebSphere MQ queues.

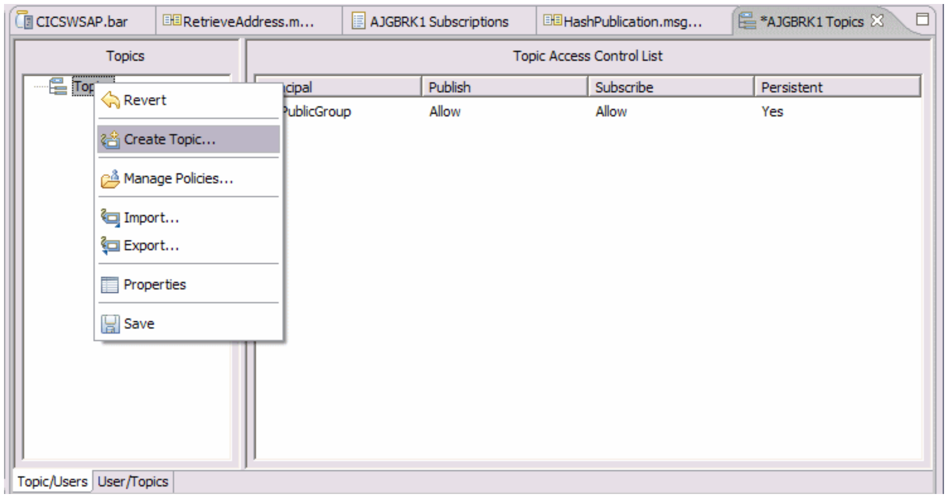
7.2 Establishing the pub/sub environment

The Change Of Address application is fundamentally a pub/sub application. As address changes are made in the postal service database via option 3 - “Add/Update Address”, notifications have to be made to all the interested parties. To achieve this end, we use WebSphere Message Broker’s pub/sub capabilities.

We want to mention the existence of the WebSphere MQ Publish/Subscribe facilities supplied as part of WebSphere MQ 5.3 fixpack 8 and above. We could certainly have used this facility as a pub/sub engine; however, WebSphere MQ Publish/Subscribe does not run on WMQ for z/OS. But apart from this, there is no technical reason we could not use the WebSphere MQ Publish/Subscribe broker by using another platform as the pub/sub engine.

It is straightforward to set up the pub/sub environment using the WMB Toolkit.

1. First, assuming your broker is up and running and the toolkit is connected, make sure the Broker Administration perspective is showing. (Windows → Open Perspective → Broker Administration perspective). See also the next section on the broker environment for more details.
2. We created a single broker called AJGBRK1 on Windows.
3. Next we create a topic called **CICSWSAP/AddressChange** that identifies our hash notification source. We open the pub/sub Topics window for this broker by double-clicking the Topics in the Domain pane at the bottom left. The toolkit now displays the topics for our broker.



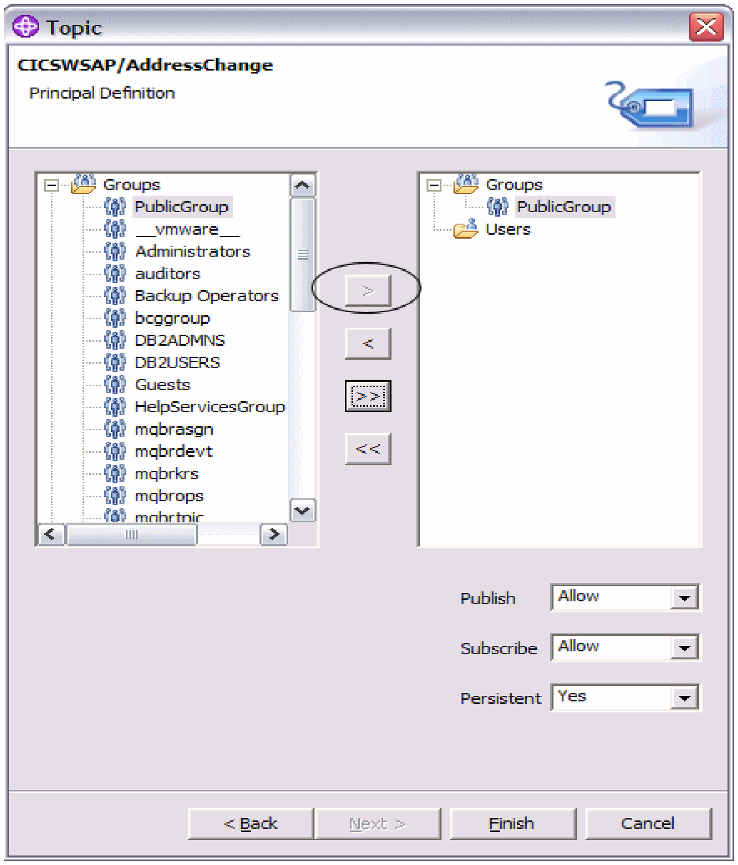
cipal	Publish	Subscribe	Persistent
ublicGroup	Allow	Allow	Yes

Figure 7-1 Create Topic 1

4. In the **Create Topic** dialogue that follows Figure 7-1, we enter our topic name: **CICSWSAP/AddressChange**, and press **Next**.

   The **Principle Definition** window now displays.

5. Since we are not too concerned about who has access to this information (address hash publications are not decodable) we select the **Public Group** group from the left pane and move it across to the right pane by pressing the "**>**" button. Figure 7-2 on page 167 is the result.



Publish: Allow
Subscribe: Allow
Persistent: Yes

Figure 7-2 Create New Topic - Principle Definition

6. Press **Finish**.

Next, we need to add a subscription into the subscription table for every client needing to be notified of the change of address. For our simple examples this is just a handful, but in reality this could be hundreds of clients. To add a subscription, WMB requires that we send a special format subscription message to a dedicated broker queue called **SYSTEM.BROKER.CONTROL.QUEUE.** The format of this message is described in the WMB Infocenter as follows:

The Register Subscriber command message is sent to a broker by a subscriber, or by another application on behalf of a subscriber, to indicate that it wants to subscribe to one or more topics at a subscription point. A message content filter can also be specified.

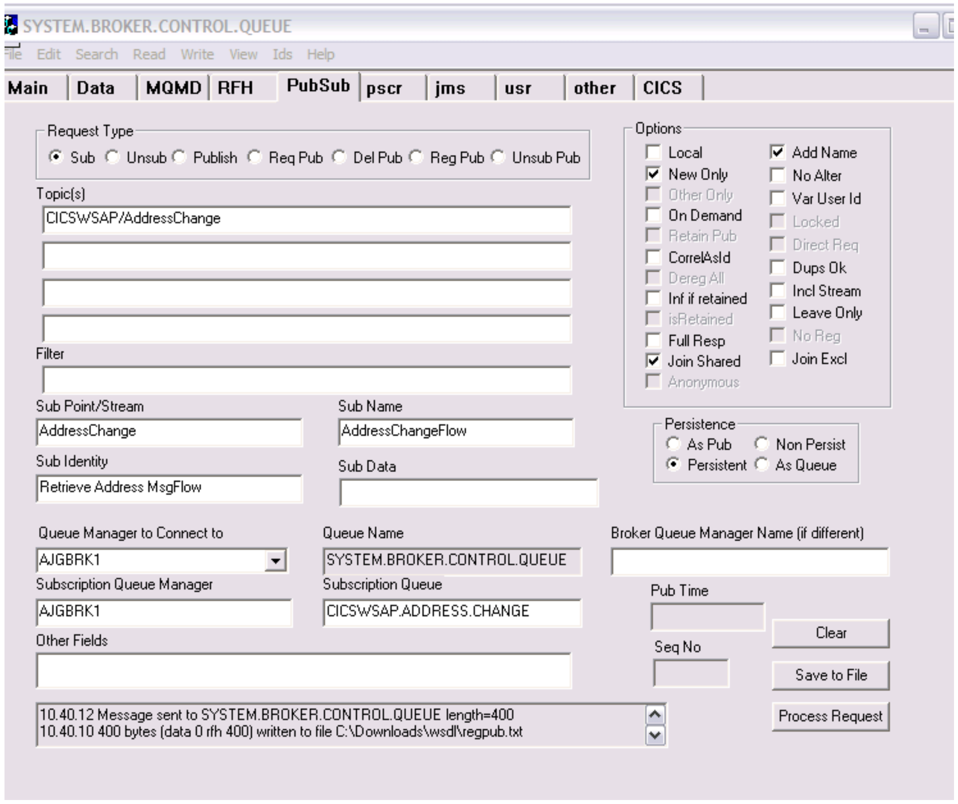
The Register Subscriber command message is an XML format message. For convenience and clarity, we use the PubSub support in the IH03 SupportPac™ - RFHUtil to create this message for us. (See the WMB Support site for a link to the supportPacs.)

1. Open RFHUtil, and click the **PubSub** tab.

   RFHUtil can construct the necessary XML command message and place the message on the broker control queue for us, but we need to supply some details about our subscription:

   – **Request type**: Subscription
   – **Topic**: CICSWSAP/AddressChange
   – **Filter**: <none>
   – **Subscription Point/Stream**: AddressChange
   – **Subscription Name**: AddressChange
   – **Subscription Identity**: Retrieve Address MsgFlow Example
   – **Subscription Queue Manager**: AJGBRK1
   – **Subscription Queue**: CICSWSAP.ADDRESS.CHANGE
   – **Options**: Join Shared, Add Name
   – **Persistence**: Persistent

   The RFHUtil window looks similar to Figure 7-3 on page 169.


SYSTEM.BROKER.CONTROL.QUEUE

File Edit Search Read Write View Ids Help
Main | Data | MQMD | RFH | PubSub | pscr | jms | usr | other | CICS

Request Type: [✓ Sub] Unsub Publish Req Pub Del Pub Reg Pub Unsub Pub

Topic(s): CICSWSAP/AddressChange

Filter:

Sub Point/Stream: AddressChange

Sub Name: AddressChangeFlow

Sub Identity: Retrieve Address MsgFlow

Sub Data:

Options: Local; [✓ New Only]; Other Only; On Demand; Retain Pub; CorrelAsId; Dereg All; Inf if retained; isRetained; Full Resp; [✓ Join Shared]; Anonymous; [✓ Add Name]; No Alter; Var User Id; Locked; Direct Req; Dups Ok; Incl Stream; Leave Only; No Reg; Join Excl

Persistence: As Pub Non Persist [✓ Persistent] As Queue

Queue Manager to Connect to: AJGBRK1

Queue Name: SYSTEM.BROKER.CONTROL.QUEUE

Broker Queue Manager Name (if different):

Subscription Queue Manager: AJGBRK1

Subscription Queue: CICSWSAP.ADDRESS.CHANGE

Pub Time:

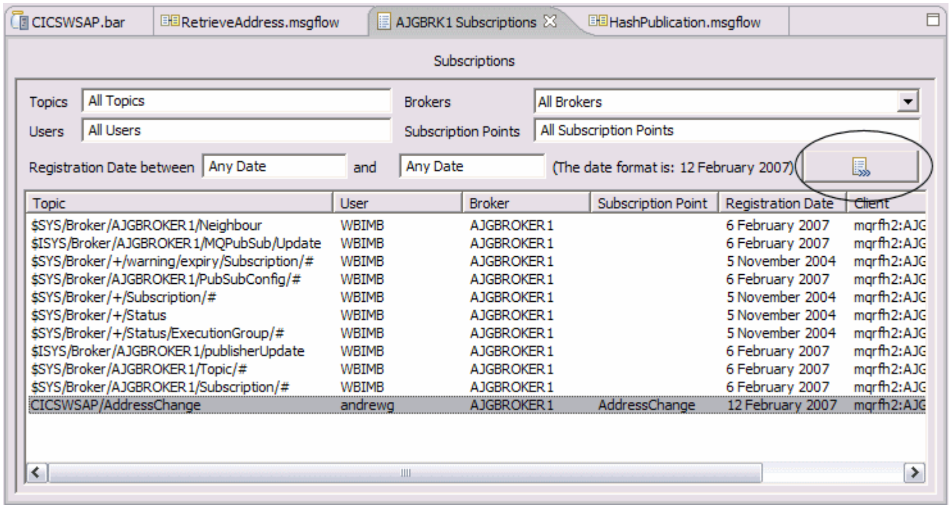
Seq No:

Other Fields:

```
10.40.12 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=400
10.40.10 400 bytes (data 0 rfh 400) written to file C:\Downloads\wsdl\regpub.txt
```

Figure 7-3 Register a subscription with RFHUtil

2. We can verify that the subscription was accepted by querying the broker's subscriptions in the toolkit.
3. Return to the broker toolkit. If not already displayed, switch to the **Broker Administration** perspective. Double-click the **Subscriptions** item in the **Domains** pane at the lower left side. This opens the broker's subscriptions.
4. By default, no filtering criteria are enforced. Click the Query button (circled in Figure 7-4 on page 170) to retrieve the registered subscriptions.



Topics: All Topics
Brokers: All Brokers
Users: All Users
Subscription Points: All Subscription Points
Registration Date between Any Date and Any Date (The date format is: 12 February 2007)

Topic	User	Broker	Subscription Point	Registration Date	Client
$SYS/Broker/AJGBROKER1/Neighbour	WBIMB	AJGBROKER1		6 February 2007	mqrfh2:AJG
$ISYS/Broker/AJGBROKER1/MQPubSub/Update	WBIMB	AJGBROKER1		6 February 2007	mqrfh2:AJG
$SYS/Broker/+/warning/expiry/Subscription/#	WBIMB	AJGBROKER1		5 November 2004	mqrfh2:AJG
$SYS/Broker/AJGBROKER1/PubSubConfig/#	WBIMB	AJGBROKER1		6 February 2007	mqrfh2:AJG
$SYS/Broker/+/Subscription/#	WBIMB	AJGBROKER1		5 November 2004	mqrfh2:AJG
$SYS/Broker/+/Status	WBIMB	AJGBROKER1		5 November 2004	mqrfh2:AJG
$SYS/Broker/+/Status/ExecutionGroup/#	WBIMB	AJGBROKER1		5 November 2004	mqrfh2:AJG
$ISYS/Broker/AJGBROKER1/publisherUpdate	WBIMB	AJGBROKER1		6 February 2007	mqrfh2:AJG
$SYS/Broker/AJGBROKER1/Topic/#	WBIMB	AJGBROKER1		6 February 2007	mqrfh2:AJG
$SYS/Broker/AJGBROKER1/Subscription/#	WBIMB	AJGBROKER1		6 February 2007	mqrfh2:AJG
CICSWSAP/AddressChange	andrewg	AJGBROKER1	AddressChange	12 February 2007	mqrfh2:AJG

Figure 7-4 Query Subscriptions in the broker toolkit

It is a now simple matter of adding new subscriptions as required using the RFHUtil.

Now, we create a simple WMB message flow to perform the message publication. The UpdateAddress application in CICS writes a message to a WMQ queue called CICSWSAP.PUBLICATION.QUEUE. Our simple publication message flow needs to take this message and publish it in our topic. Next, we deal with creating this message flow in WMB.

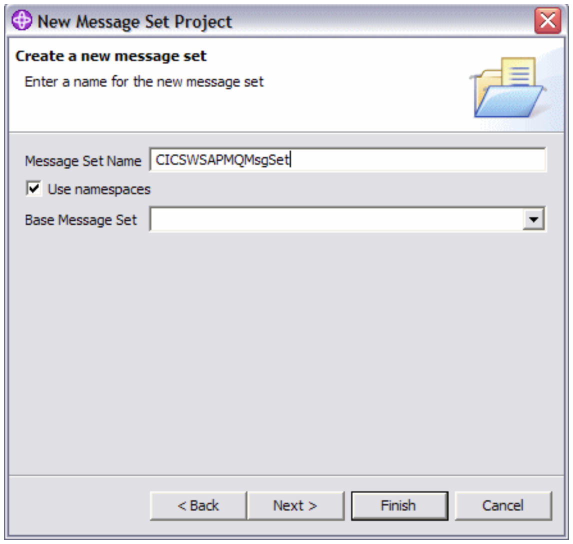
7.3 Creating the Hash Notification Message Set

Before the broker can perform any work on messages we choose to send it, we need to tell it about the formats of these messages. We need the broker to be able to handle the following two types of messages:

- Simple 12-character strings: the hash notifications
- The WSDL-defined Web-service messages for our GetHash and RetrieveAddress CICS web-services

We need to create a Message Set project to contain these message definitions, but it is best practice to separate different types of messages into separate projects; therefore, we will create two Message Set projects: one for the hash notification messages and another for the WSDL-defined messages. We cover the WSDL message set in section“Creating a Message Set from the RetrieveAddress WSDL” on page 197.

1. Switch to the **Broker Application Development** perspective.
2. Select File → New → Message Set Project. We call our CICSWSAPMQMsgSet, which reflects the origin of the message. Figure 7-5 appears:

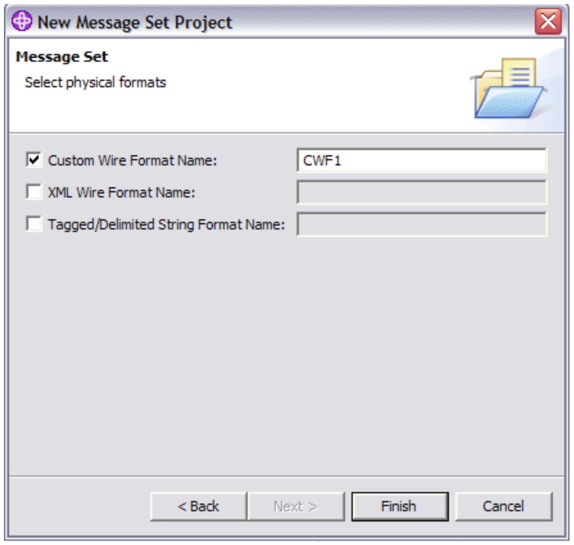


Message Set Name: CICSWSAPMQMsgSet
☑ Use namespaces

Figure 7-5 Create a new Message Set Project

Creating the Message Set project also creates a Message Set. Here we get to name the message set, and set **Use Namespaces** flag. We kept the Message Set name the same as the Message Project name.

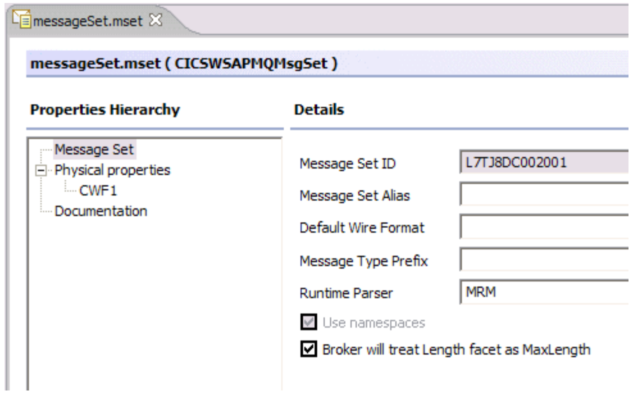
Figure 7-6 defines the type of messages contained in this message set and in theory the whole project. For the simple, 12-char string type messages, we use the CWF format.



[✓] Custom Wire Format Name: CWF1
[ ] XML Wire Format Name:
[ ] Tagged/Delimited String Format Name:

Figure 7-6 Set the Physical Format of the Message Set

3. Press finish to create the Message Set Project, the Message Set, and to open the Message Set editor.

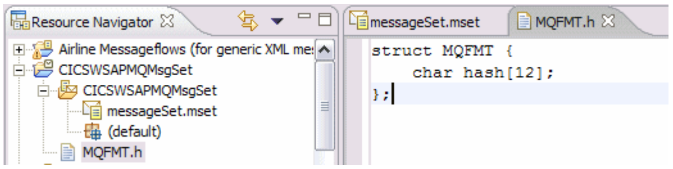


Message Set ID: L7TJ8DC002001
Message Set Alias:
Default Wire Format:
Message Type Prefix:
Runtime Parser: MRM
[✓] Use namespaces
[✓] Broker will treat Length facet as MaxLength

Figure 7-7 The Message Set editor

4. Next, we need to define the format of our simple, 12-char string format in this message set. While this could be done manually, it is better to import a language structure such as a COBOL CopyBook or C Structure and let the WMB broker do the work. In our case, we have no structure defined for this

message in our application. We just MQPUT a 12-char string onto CICSWSAP.PUBLICATION.QUEUE. However, it is a trivial matter to create a C structure to map this message. To do this we create a C header file in the Broker toolkit under the CICSWSAPMQMsgSet project (File → New → Other → Simple → File: MQFMT.h) This invokes a simple text editor window into which we can place our C structure, as seen in Figure 7-8.

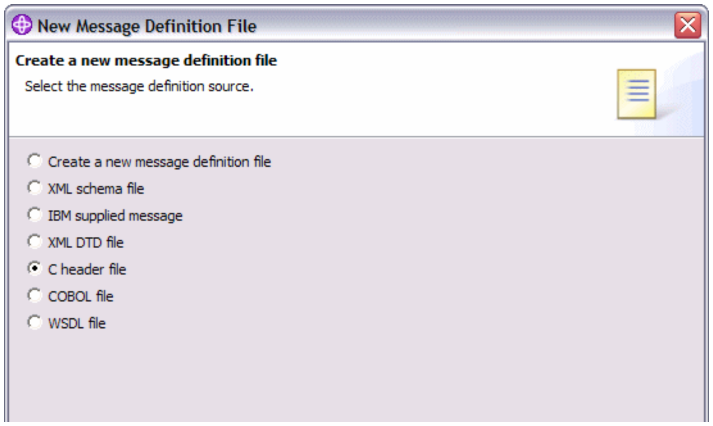


```
struct MQFMT {
      char hash[12];
};
```

Figure 7-8 Define a C Header for the MQ Message Format

Save this file.

5. Now we must import this file into our CICSWSAPMQMsgSet message set. We do this by creating a new Message Definition File (New → Message Definition File).

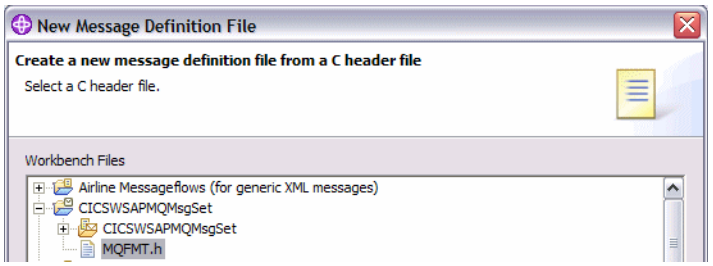


Create a new message definition file
Select the message definition source.

- ( ) Create a new message definition file
- ( ) XML schema file
- ( ) IBM supplied message
- ( ) XML DTD file
- [✓ C header file]
- ( ) COBOL file
- ( ) WSDL file

Figure 7-9 Import the C Header file

6. Next, we must indicate the location of the C Header file, which is what we just created, as shown in Figure 7-10 on page 174.


New Message Definition File

Create a new message definition file from a C header file

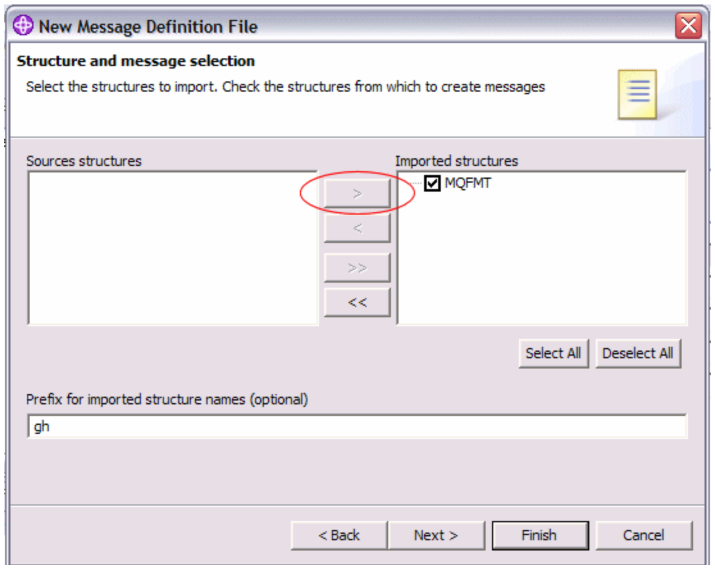
Select a C header file.

Workbench Files

- Airline Messageflows (for generic XML messages)
- CICSWSAPMQMsgSet
  - CICSWSAPMQMsgSet
  - MQFMT.h

Figure 7-10 Select the C Header file

7. Select the MQFMT in the source structure, then press the “>” button. Check the MQFMT in the imported structures.


New Message Definition File

Structure and message selection

Select the structures to import. Check the structures from which to create messages

Sources structures

Imported structures

☑ MQFMT

Prefix for imported structure names (optional)

gh

Figure 7-11 Complete the C header input

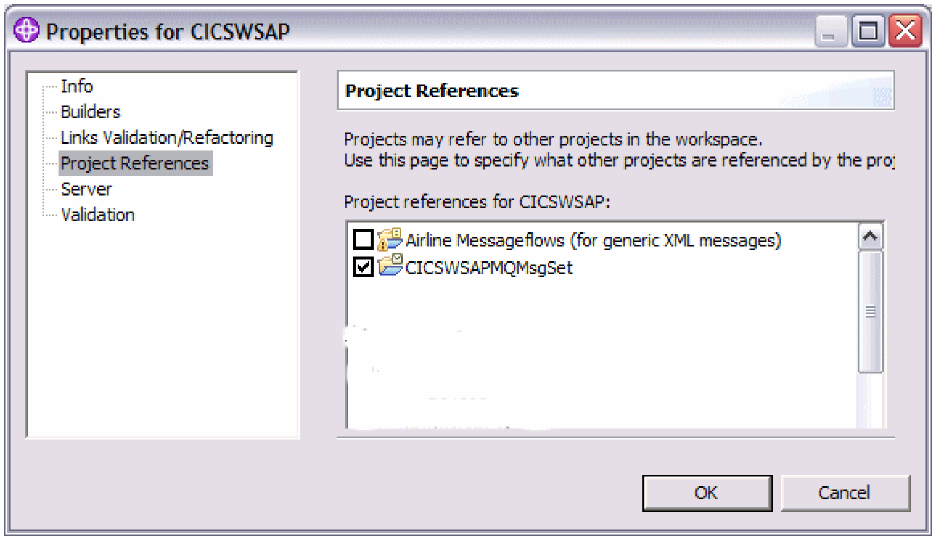
8. We chose to set the prefix for the structure to “gh”, which stands for GetHash, which reflects the function that generated the hash code. We now click the **Finish** button to complete the import.

At this point, we created a definition so that WMB can understand the content of the message being taken from our publication queue. Next we must create very simple Message Flow to do the actual publication to the CICSWSAP/AddressChange topic.

7.4 Creating the Publication Message Flow

As mentioned in the WMB overview, Message Flows are the way we program WMB. Message flows are sequences of Nodes that perform functions, connected together.

The message flow needs to be defined in a separate **Message Flow** project. (File → New → Message Flow Project). We need to specify the dependencies this project has on other projects that contain message definitions we require, specifically the CICSWSAPMQMsgSet project.



Projects may refer to other projects in the workspace.
Use this page to specify what other projects are referenced by the proj

Project references for CICSWSAP:

- [ ] Airline Messageflows (for generic XML messages)
- [✓] CICSWSAPMQMsgSet

Figure 7-12 Message Flow project references

Our publication message flow will be very straightforward. We simply need to take a publication message of the CICSWSAP.PUBLICATION.QUEUE queue and publish it. In order to publish the message, WMB first needs to be able to understand the message. This is why we needed to create the CICSWSAPMQMsgSet message set and import the structure. We used the following message flow to perform our publication.

We created the Message Flow project and now a Message Flow called HashPublication. With this, the message flow editor opens with a blank canvas.

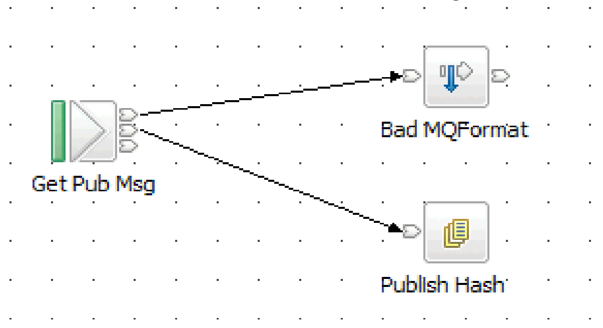
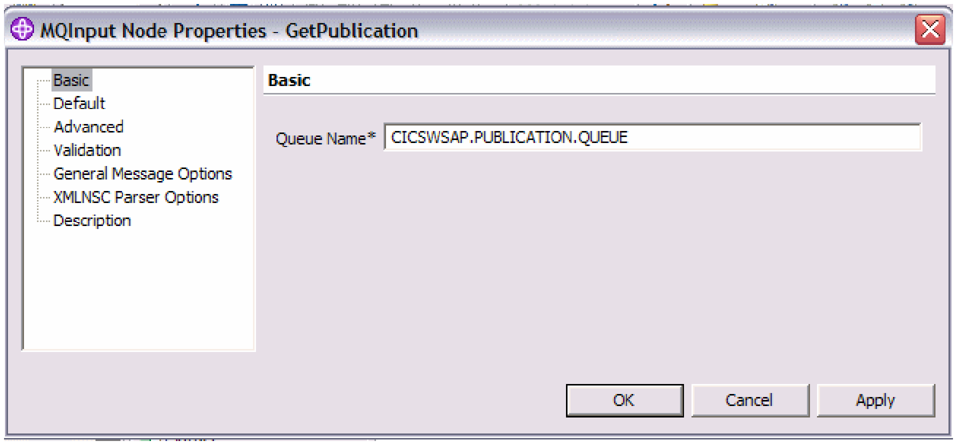


Figure 7-13 The Hash Publication Message Flow

Construction of this Message Flow is a simple matter of dropping the required nodes onto the canvas, connecting the nodes, and renaming nodes as appropriate. Of the three terminals of the MQInput node (Get Pub Msg), we have the **failure** terminal connected to the Trace node named BadMQFormat. The trace node is configured to write the message tree out to the local error log, which is viewed using the EventVwr utility in Windows. This gives us a chance to observe erroneous messages passed from the UpdateAddress CICS application.

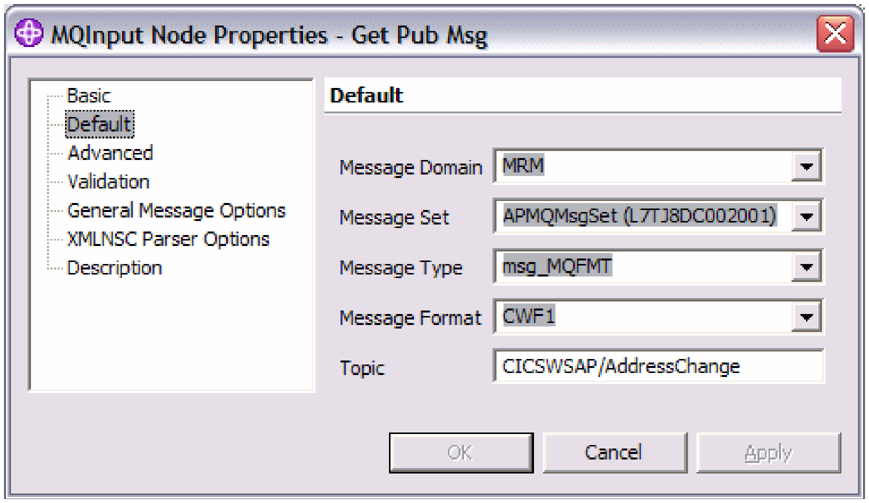
The MQInput's **Out** terminal is connected to the Publication node (Publish Hash).

The MQInput node has the basic properties shown in Figure 7-14 and the default properties shown in Figure 7-15 on page 177.



Queue Name* CICSWSAP.PUBLICATION.QUEUE

Figure 7-14 MQInput Node - Basic Properties


MQInput Node Properties - Get Pub Msg

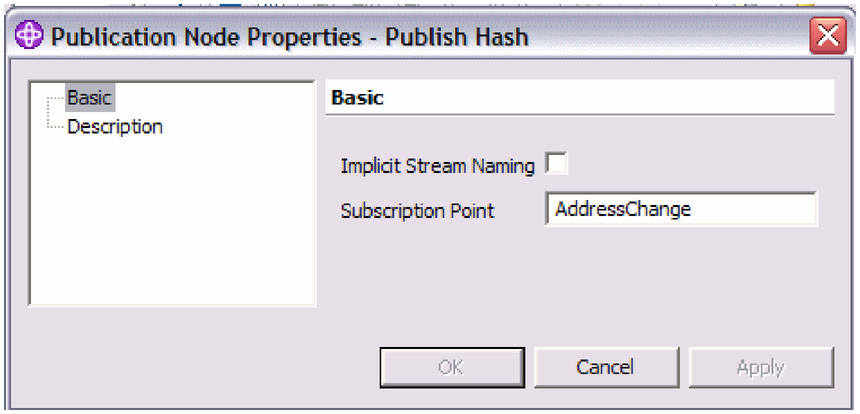
Default

Message Domain: MRM
Message Set: APMQMsgSet (L7TJ8DC002001)
Message Type: msg_MQFMT
Message Format: CWF1
Topic: CICSWSAP/AddressChange

Figure 7-15 The MQInput node - Default Properties

Each of these properties are essential to the correct operation of the MQInput node and subsequent publication of the message. We must tell the MQInput node from what queue to take messages. We must indicate the message set that tells WMB about the format of these messages. This format is the one we created by importing the C header file containing the C structure. The other piece of critical information is to define for what pub/sub topic this message is destined.

The Publication node contains just one configuration item—the Subscription point.


Publication Node Properties - Publish Hash

Basic

Implicit Stream Naming: [ ]
Subscription Point: AddressChange

Figure 7-16 The Publication node properties

We deployed this message flow, together with the CICSWSAPMQMsgSet—termed a *dictionary* in the broker archive (BAR) file. Assuming required queues are defined, channels running and subscriptions made, our pub/sub environment is now ready.

7.5 Testing the Publication Notification Message Flow

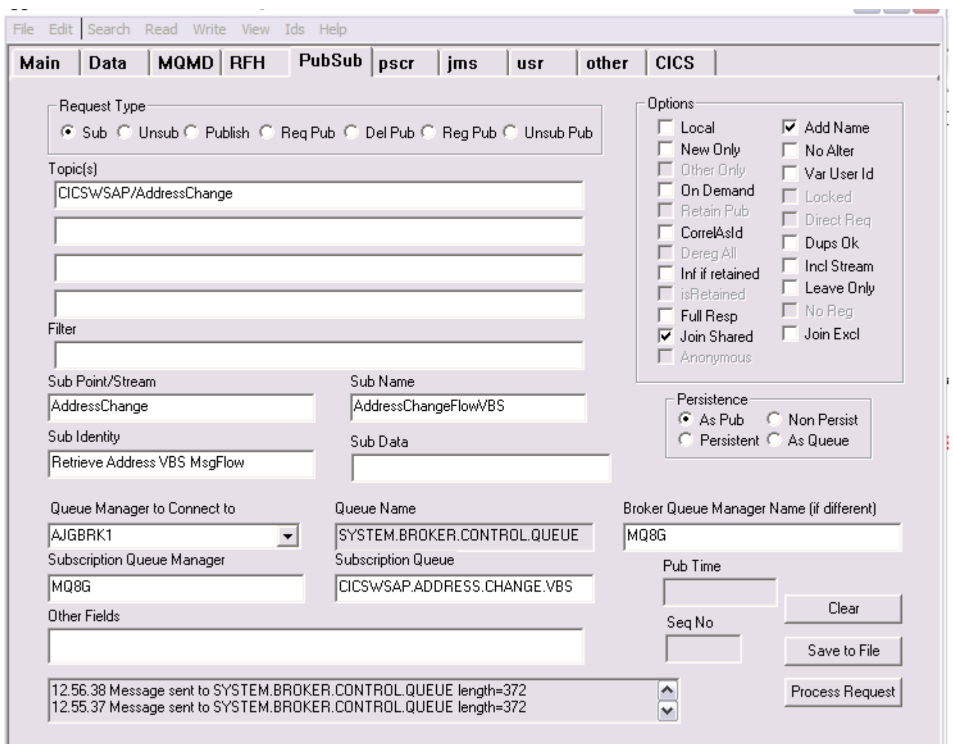
To test our pub/sub configuration, even without the presence of any actual subscribers, we simply place a message on the CICSWSAP.PUBLICATION.QUEUE, providing it is in the correct format. We could then check the subscriber queue (for example, CICSWSAP.ADDRESS.CHANGE) for the presence of the message. If the message is not the expected format, a trace entry is placed in the local error log indicating the error.

7.6 Porting the Publication Notification Message Flow to System z Broker

No material changes are required to have the pub/sub message flow run on a System z broker. Of course, the **CICSWSAP/AddressChange** publication needs to be registered against the System z broker, which works exactly the same way as the Windows broker.

Registering subscriptions is slightly different because we need to write to a remote queue. Staying with RFHUtil, we have two options.

1. Use RFHUTILC - a WMQ Client version of RFHUTIL that is included in the IH03 supportpac. This facilitates connecting to the System z WMQ manager MQ8G as a WMQ client. We then write our subscription message to MQ6G’s local SYSTEM.BROKER.CONTROL.QUEUE exactly the way we did for the Windows broker.
2. Use RFHUTIL, specifying a remote queue. This requires that our local queue manager AJGBRK1 has functioning channels to the System z queue manager MQ8G. It is then just a case of specifying the remote parameters, as shown in Figure 7-17 on page 179.


File Edit Search Read Write View Ids Help
Main Data MQMD RFH PubSub pscr jms usr other CICS

Request Type: [✓ Sub] Unsub Publish Req Pub Del Pub Reg Pub Unsub Pub

Topic(s): CICSWSAP/AddressChange

Filter:

Sub Point/Stream: AddressChange

Sub Name: AddressChangeFlowVBS

Sub Identity: Retrieve Address VBS MsgFlow

Sub Data:

Options: Local, New Only, Other Only, On Demand, Retain Pub, CorrelAsId, Dereg All, Inf if retained, isRetained, Full Resp, [✓ Join Shared], Anonymous, [✓ Add Name], No Alter, Var User Id, Locked, Direct Req, Dups Ok, Incl Stream, Leave Only, No Reg, Join Excl

Persistence: [✓ As Pub] Non Persist Persistent As Queue

Queue Manager to Connect to: AJGBRK1

Queue Name: SYSTEM.BROKER.CONTROL.QUEUE

Broker Queue Manager Name (if different): MQ8G

Subscription Queue Manager: MQ8G

Subscription Queue: CICSWSAP.ADDRESS.CHANGE.VBS

Pub Time:

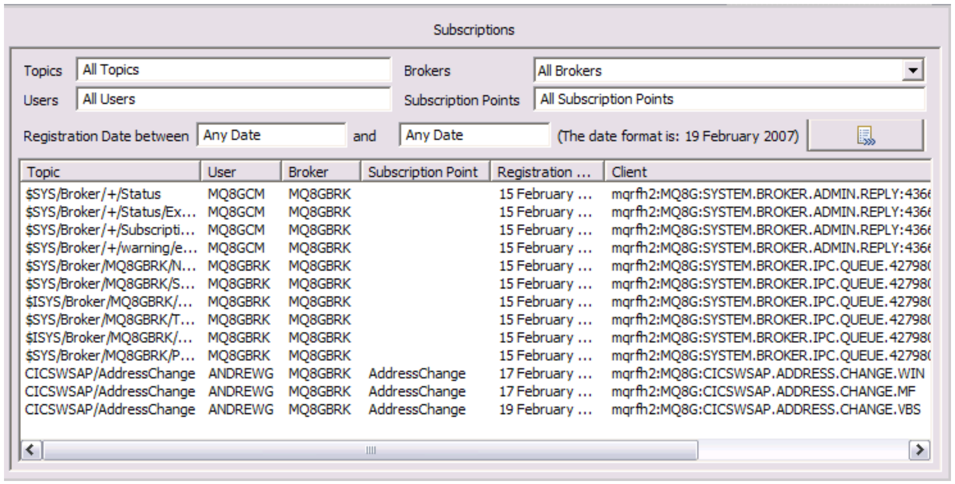
Seq No:

Other Fields:

```
12.56.38 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=372
12.55.37 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=372
```

Figure 7-17 Register a Subscription for the System z broker

Subscriptions now appear against the MQ8G broker, as shown in Figure 7-18 on page 180.



Topic	User	Broker	Subscription Point	Registration …	Client
$SYS/Broker/+/Status	MQ8GCM	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.ADMIN.REPLY:436
$SYS/Broker/+/Status/Ex…	MQ8GCM	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.ADMIN.REPLY:436
$SYS/Broker/+/Subscripti…	MQ8GCM	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.ADMIN.REPLY:436
$SYS/Broker/+/warning/e…	MQ8GCM	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.ADMIN.REPLY:436
$SYS/Broker/MQ8GBRK/N…	MQ8GBRK	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.IPC.QUEUE.42798
$SYS/Broker/MQ8GBRK/S…	MQ8GBRK	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.IPC.QUEUE.42798
$ISYS/Broker/MQ8GBRK/…	MQ8GBRK	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.IPC.QUEUE.42798
$SYS/Broker/MQ8GBRK/T…	MQ8GBRK	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.IPC.QUEUE.42798
$ISYS/Broker/MQ8GBRK/…	MQ8GBRK	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.IPC.QUEUE.42798
$SYS/Broker/MQ8GBRK/P…	MQ8GBRK	MQ8GBRK		15 February …	mqrfh2:MQ8G:SYSTEM.BROKER.IPC.QUEUE.42798
CICSWSAP/AddressChange	ANDREWG	MQ8GBRK	AddressChange	17 February …	mqrfh2:MQ8G:CICSWSAP.ADDRESS.CHANGE.WIN
CICSWSAP/AddressChange	ANDREWG	MQ8GBRK	AddressChange	17 February …	mqrfh2:MQ8G:CICSWSAP.ADDRESS.CHANGE.MF
CICSWSAP/AddressChange	ANDREWG	MQ8GBRK	AddressChange	19 February …	mqrfh2:MQ8G:CICSWSAP.ADDRESS.CHANGE.VBS

Figure 7-18 Subscriptions on MQ8GBRK

8

Developing Web service clients

In this chapter we demonstrate how to create a number of different Web service clients to perform the client side of our Change of Address application. These range from the very simple—in the case of the WebSphere Application Server client generated from WSDL using a wizard—to the more elaborate using WebSphere Message Broker. In between, we show a client developed using VBScript and the WebSphere MQ Client software.

Of course, these examples are not exhaustive. Virtually any software product that can call a Web service and be invoked by a WMQ Client trigger is a suitable candidate for this task. We also wanted to show that we do not limit the client side of our application to a strict set of steps.

8.1 Using VB Script

VBScript (short form of Microsoft Visual Basic® Scripting Edition) is a scripting language and is based on Microsoft's Visual Basic. It can be executed as a standalone application, or, as in our example, it can be embedded in a Web page.

Here we present two scripts, the first to invoke a check for a change of address and the second provides a report of all corporate clients who acknowledged receipt of the change of address notification.

8.1.1 VBScript Retrieve Address Query

The intention was to demonstrate how a simple Web page, (with embedded VBScript) could be used to implement our Query Change of Address Request. We wanted to have a Web page with a command button that when clicked would check WMQ for a change of address notification. If one existed it would retrieve the data and then use the data to invoke the RetrieveAddress CICS Web service to retrieve the new address details.

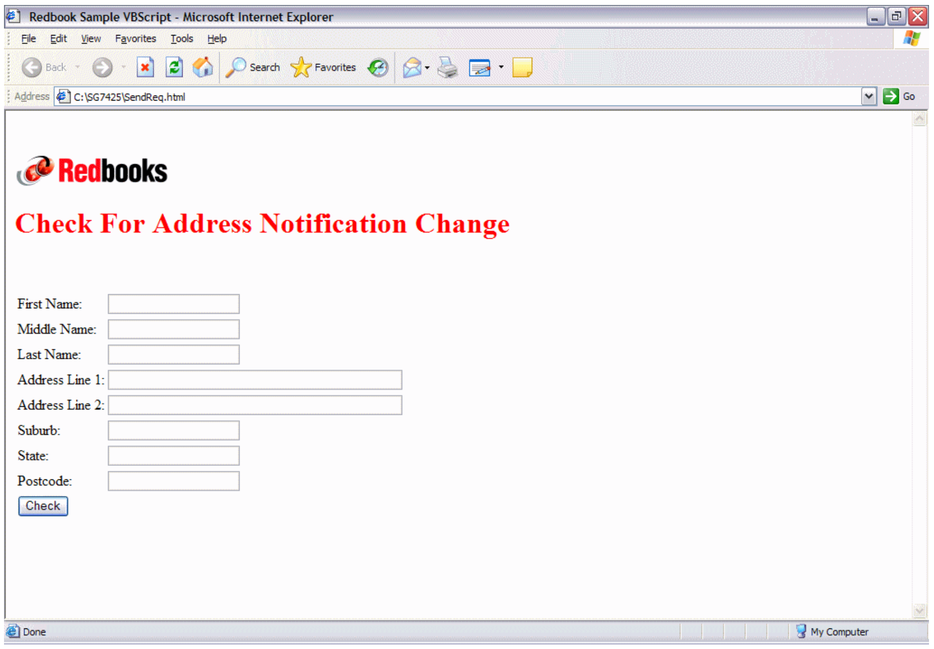
In order to run the application we required the following:

- **IBM WebSphere MQ Client,** which we downloaded from the following Web address:

  `http://www.ibm.com/support/us/`

- An HTML file containing the required VBScript code. The contents of this file, RetAddr.html, are shown in Example 8-1 on page 184. This information is also available in Appendix A, "Additional material" on page 249.

When the HTML file was displayed in a Web browser it gave the panel shown in Figure 8-1 on page 183.


Redbook Sample VBScript - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
C:\SG7425\SendReq.html
Go
Redbooks
Done
My Computer

Check For Address Notification Change

First Name:
Middle Name:
Last Name:
Address Line 1:
Address Line 2:
Suburb:
State:
Postcode:
Check

Figure 8-1 VBScript embedded in an HTML page to invoke Web service

8.1.2 VBScript code overview

When the *Check* button is clicked the **cmdSubmit_OnClick** subroutine is invoked. The first call made by this code is to the **MQGet** function, which makes the WMQ calls to check and retrieve any data from the relevant queue.

If data is returned from **MQGet**, the cmdSubmit_OnClick subroutine then uses this data to build a Web service request to query the new address details. The request is built based on the wsdl file generated from the DFHLS2WS invocation for the ITSORA03 source.

Having received the new address details from the CICS Web services call, the code parses it and displays the results in the corresponding fields on the Web page.

It should be noted that the VBScript code is not particularly robust and makes certain assumptions. We preferred taking this approach to writing a more robust version that would detract from the core aspects that we are trying to demonstrate.

Example 8-1 Source code for RetAddr.htm

```
<HTML>
<HEAD>
<TITLE>Redbook Sample VBScript</TITLE>
<SCRIPT LANGUAGE="VBScript">
Option Explicit
```

Function MQGet()

```
' This function is loosly based on the sample, mqaxtriv.htm, provided with MQ client V6.
' It simply issues a get and if successful passes the data, a AddressHash value,
' back to the caller who in turn uses this to make a CICS Web Services call to get the
' Updated address.

Dim qm, Queue, q, qmanager

Dim gmsg       ' message for getting
Dim gmo
Dim gs

Dim cr  ' carriage return
  cr = chr(13)
  MQGet = ""

  on error resume next


  qmanager = "AJGBRK1"
  Set qm = MQAXSession.AccessQueueManager(qmanager)

  if err <> 0 then
    if err = 438 then
      msgbox "Cannot run example.  Please check:" _
             & cr & "    security settings (see text of this html page)" _
             & cr & "    proper installation and registration of MQAX."
    elseif err = 32000 then
        msgbox "Cannot access WebSphere MQ queue manager:" _
              & cr & "    " & MQAXSession.reasonName
    else
      msgbox "Unexpected error in MQGet " & err.description
    end if
    exit function
  end if

  ' access a standard queue that should be there
  Set q = qm.AccessQueue("CICSWSAP.ADDRESS.CHANGE.VBS", 16 or 1) ' MQOO_OUTPUT Or
MQOO_INPUT_AS_Q_DEF)


  Set gmsg = MQAXSession.AccessMessage()

  ' create a (default) MqGetMessageOptions object
  Set gmo = MQAXSession.AccessGetMessageOptions()

  ' do the get
  q.Get gmsg, gmo

  if err <> 0 then
    if MQAXSession.reasoncode = 2033 then
      ' msgbox "No item on queue:" & cr & "    " & MQAXSession.reasonName
    else
      msgbox "Unexpected error in MQGet Function " & err.description
    end if
    exit function
  end if
  ' now read the data from the input message
  gs = gmsg.ReadString(gmsg.MessageLength)

  MQGet = gs

End Function
```

Sub cmdSubmit_OnClick

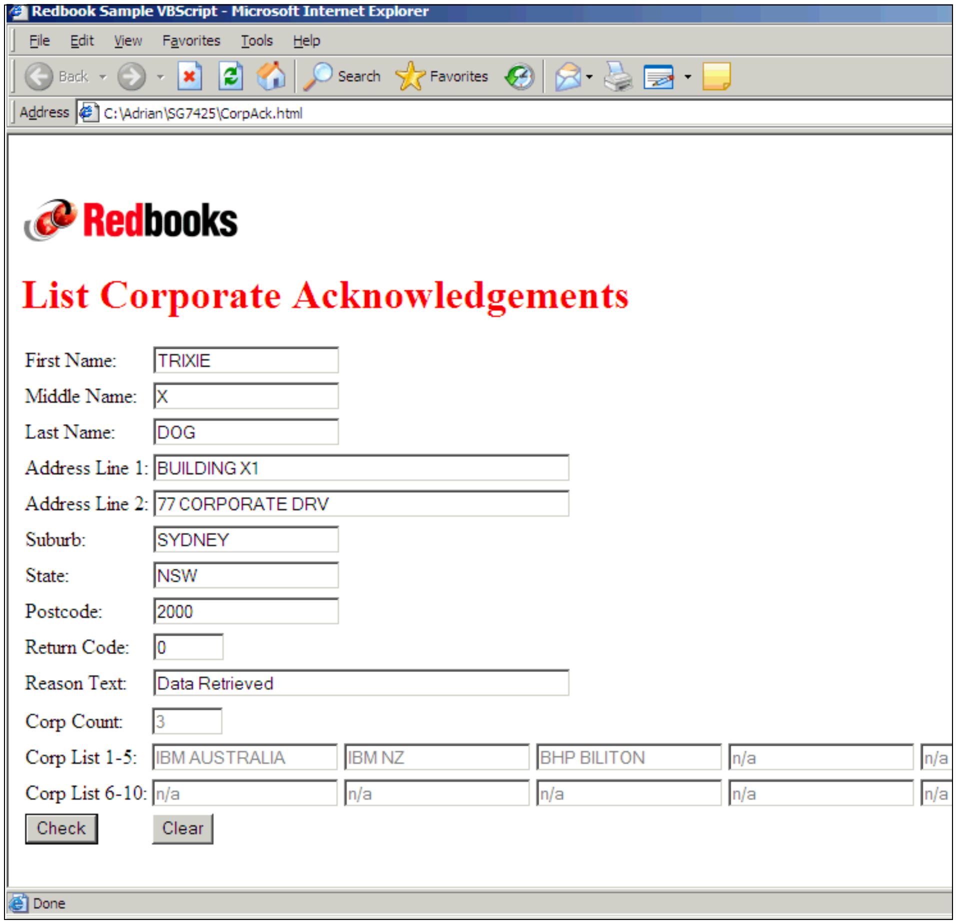
```
Dim Hash
Dim pos, pos2, StartTag, EndTag
Dim xmlhttp, strRequestFilePath, strURI, xmlRequest, strRequest

Const conHostName = "WTSC66.ITSO.IBM.COM"
Const conPortNumber = "8012"
Const conWebServicePath = "cicswsap/RetrieveAddress"


  document.frmExample5a.txtFirstName.value = " "
  document.frmExample5a.txtMiddleName.value = " "
  document.frmExample5a.txtAddressLine1.value = " "
  document.frmExample5a.txtAddressLine2.value = " "
  document.frmExample5a.txtSuburb.value = " "
  document.frmExample5a.txtState.value = " "
  document.frmExample5a.txtPostcode.value = " "

  ' See if there is an AddressHash value on the queue.
  Hash = MQGet()
  If Hash = "" Then
    MsgBox "No item on queue"
    Exit Sub
  End If



  ' We have a AddressHash value to invoke web service to get address.
  ' Create the object that will send the API request and received the API response
  Set xmlhttp = CreateObject("Microsoft.XMLHTTP")

  ' Set up the URL to the CICS server
    strURI = "http://" & conHostName & ":" & conPortNumber & "/" & conWebServicePath

  ' Initialize the HTTP request
  xmlhttp.open "POST", strURI, False

  ' Create the request header

  xmlhttp.setRequestHeader "Content-Type", "text/xml; charset=""UTF-8"""
  xmlhttp.setRequestHeader "Host", conHostName & ":" & conPortNumber
  xmlhttp.setRequestHeader "Connection", "Keep-Alive"

  ' In theory having retrieved a AddressHash value the application should look this up
  ' on a 'local' database to retrieve the corresponding name. For simplicity
  ' at this point we will just hard code the name.


  ' Send the request to the Web service

  strRequest =  "<?xml version=""1.0"" encoding=""UTF-8"" ?>" _
            & "<SOAP-ENV:Envelope xmlns:SOAP-ENV=""http://schemas.xmlsoap.org/soap/envelope/"">" _
            & "<SOAP-ENV:Header SOAP-ENV:mustUnderstand=""no""/>" _
            & "   <SOAP-ENV:Body>                                  " _
            & "         <ITSORA03Operation>                        " _
            & "           <raca>                                   " _
            & "             <FirstName>ADRIAN</FirstName>          " _
            & "             <MiddleName></MiddleName>              " _
            & "             <LastName>SIMCOCK</LastName>           " _
            & "             <AddressLine1></AddressLine1>           " _
            & "             <AddressLine2></AddressLine2>           " _
            & "             <Suburb></Suburb>                       " _
            & "             <State></State>                         " _
            & "             <Postcode></Postcode>                   " _
            & "             <AddressHash>" & Hash & "</AddressHash> " _
            & "             <ClientId>1</ClientId>                  " _
            & "             <NameRef>4</NameRef>                    " _
            & "             <rc>0</rc>                              " _
            & "             <reason> </reason>                      " _
            & "           </raca>                                   " _
            & "         </ITSORA03Operation>                        " _
            & "     </SOAP-ENV:Body>                                " _
            & "   </SOAP-ENV:Envelope>                              "



  xmlhttp.send strRequest

  If Err.Number Then
    MsgBox "Could not send command. CICS region might be down.", vbExclamation
  Else
    ' In theory we now have the response data from CICS so we will parse it up
    ' and display it to the user.

    StartTag = "<FirstName>"
    EndTag = "</FirstName>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtFirstName.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if

    StartTag = "<MiddleName>"
    EndTag = "</MiddleName>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtMiddleName.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if

    StartTag = "<LastName>"
    EndTag = "</LastName>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtLastName.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if

    StartTag = "<AddressLine1>"
    EndTag = "</AddressLine1>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtAddressLine1.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if

    StartTag = "<AddressLine2>"
    EndTag = "</AddressLine2>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtAddressLine2.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if

    StartTag = "<Suburb>"
    EndTag = "</Suburb>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtSuburb.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if

    StartTag = "<State>"
    EndTag = "</State>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtState.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if

    StartTag = "<Postcode>"
    EndTag = "</Postcode>"
    pos = Instr(xmlhttp.responseText,StartTag)
    if pos > 0 then
      pos2 = Instr(xmlhttp.responseText,EndTag)
      document.frmExample5a.txtPostcode.value =
mid(xmlhttp.responseText,pos+Len(StartTag),(pos2-(pos+Len(StartTag))))
    End if
  End If
End Sub
</SCRIPT>
</HEAD>
<BODY>
<!-- my test Mq object (guid for MQAXSessionession) -->
<OBJECT
    classid="clsid:00290471-B893-11CF-A5F7-444553540000"
    id=MQAXSession
>
</OBJECT>
<p><img src="redbook.gif" width="180" height="33" alt="Redbook">
<font color="red">
<H1>Check For Address Notification Change</H1>
&nbsp;
</font>

<FORM NAME="frmExample5a">
  <TABLE>

    <TR>
      <TD>First Name:</TD>
      <TD><INPUT TYPE="Text" NAME="txtFirstName" SIZE="20" disabled></TD>
    </TR>
    <TR>
      <TD>Middle Name:</TD>
      <TD><INPUT TYPE="Text" NAME="txtMiddleName" SIZE="20" disabled></TD>
    </TR>
    <TR>
      <TD>Last Name:</TD>
      <TD><INPUT TYPE="Text" NAME="txtLastName" SIZE="20" disabled></TD>
    </TR>
    <TR>
      <TD>Address Line 1:</TD>
      <TD><INPUT TYPE="Text" NAME="txtAddressLine1" SIZE="50" disabled></TD>
    </TR>
    <TR>
```

```
      <TD>Address Line 2:</TD>
      <TD><INPUT TYPE="Text" NAME="txtAddressLine2" SIZE="50" disabled></TD>
    </TR>
    <TR>
      <TD>Suburb:</TD>
      <TD><INPUT TYPE="Text" NAME="txtSuburb" SIZE="20" disabled></TD>
    </TR>
    <TR>
      <TD>State:</TD>
      <TD><INPUT TYPE="Text" NAME="txtState" SIZE="20" disabled></TD>
    </TR>
    <TR>
      <TD>Postcode:</TD>
      <TD><INPUT TYPE="Text" NAME="txtPostcode" SIZE="20" disabled></TD>
    </TR>
    <TR>
      <TD><INPUT TYPE="Button" NAME="cmdSubmit" VALUE="Check"</TD>
      <TD></TD>
    </TR>
  </TABLE>
  </FORM>
</BODY>
</HTML>
```

8.1.3 VBScript Corporate Acknowledgement Query

This second VBScript runs in much the same was as the first. This time we call the Corporate Acknowledgements CICS Web service to retrieve the audit records of all the corporate clients who received the change of address notification and then retrieved the new address. Although this does not guarantee that the external client successfully updated their customer database, it is a close approximation.

Unlike the rest of the CICSWSAP application suite, the ITSOCA03 code is CICS/DB2 COBOL. This difference presents no difficulties either in Web service generation or invocation. In keeping with the Web service philosophy, the client side has no knowledge of the implementation details of the server side. So this makes no difference.


Redbook Sample VBScript - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Adrian\SG7425\CorpAck.html
Redbooks

List Corporate Acknowledgements

First Name:	TRIXIE				
Middle Name:	X				
Last Name:	DOG				
Address Line 1:	BUILDING X1				
Address Line 2:	77 CORPORATE DRV				
Suburb:	SYDNEY				
State:	NSW				
Postcode:	2000				
Return Code:	0				
Reason Text:	Data Retrieved				
Corp Count:	3				
Corp List 1-5:	IBM AUSTRALIA	IBM NZ	BHP BILITON	n/a	n/a
Corp List 6-10:	n/a	n/a	n/a	n/a	n/a

Figure 8-2 List Corporate Acknowledgements

8.2 Generating a Java Client using WAS

This section demonstrates the use of WebSphere Developer for System z (WD/z) to generate a Web service client and run it in an application server. WD/z generates all the required Java classes to create a Web service request and to receive a Web service response. It also builds a very basic graphical user interface to interact with those classes. RAD users will be able to take advantage of this facility as the wizard is part of the RAD Web service features. As WD/z is built on RAD, this facility is also available under WD/z.

For this example, we generate a client to invoke the GetHash Web service. The only input required is the GetHash.wsdl.

Import the wsdl

The wsdl needs to be in the WD/z work space. If you do not already have a local copy of the wsdl, import the wsdl using **File** → **Import**, and then select the import source. Follow the wizard to complete the import.

Start the application server

Start the server before we start to generate the Web client using the wizard because it can take several minutes to start the WebSphere Application Server.

Note: WebSphere Application Server v6.1 comes bundled as an optional product in IBM Rational Application Developer V7 or RAD V7, which is a pre-requisite for running Web service clients under WD/z or RAD.

To start the server, select it in the Servers view. Go to **Window** → **Show View** → **Other**, and select **Server** → **Servers**, and click **OK**. This is seen in Figure 8-3 on page 191.

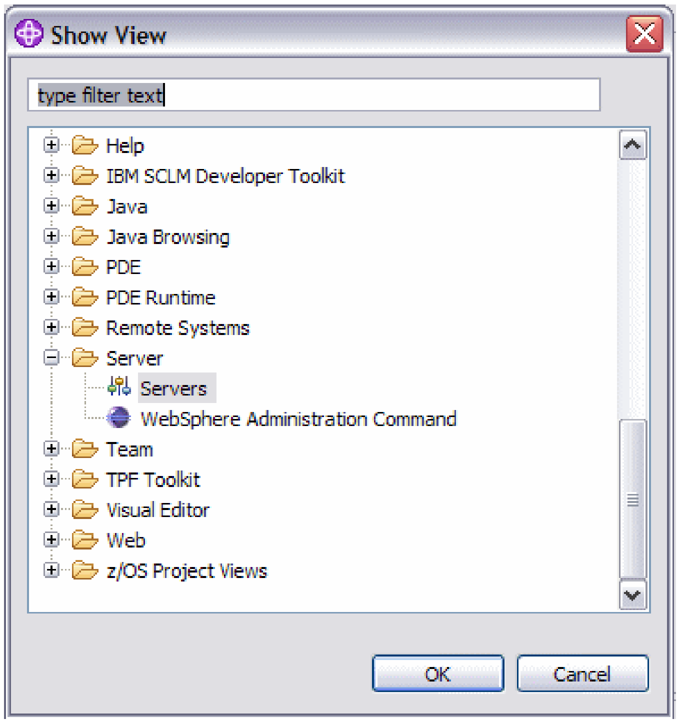


Figure 8-3 Select the Server view

In the Servers view, right-click the server name, and select **Start**. This begins the server startup, and progress can be seen in the console view that is dynamically opened. The server view shows the status as in Figure 8-4.

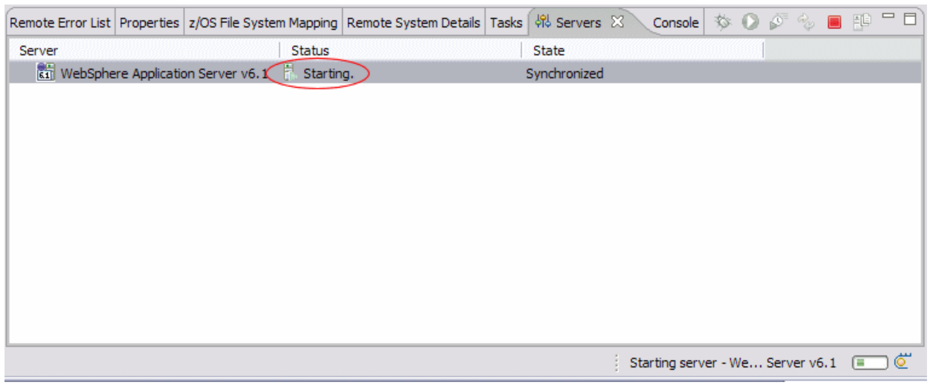


Figure 8-4 The Server view showing startup

A successful startup of the server will show a State of **Started** and a Status of **Synchronized**. The Console view shows the following message:

```
WSVR0001I: Server server1 open for e-business
```

Generate the client

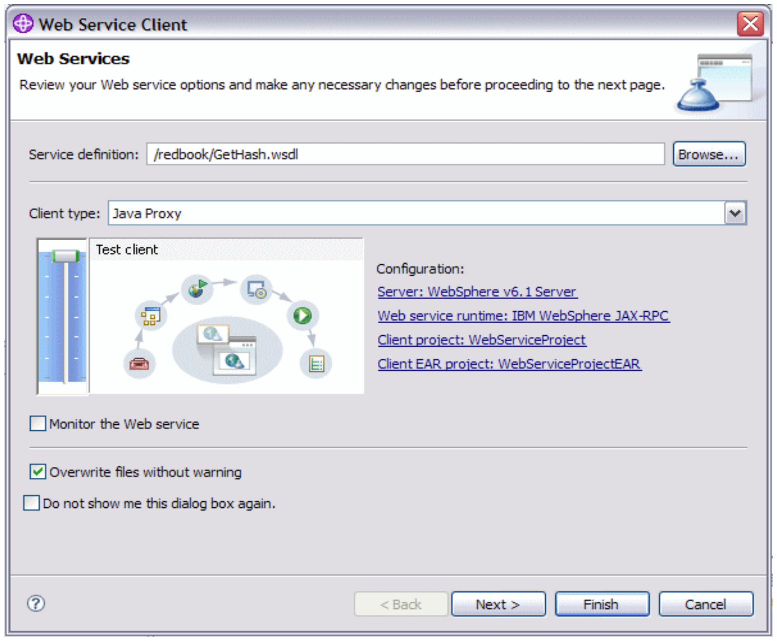
We are now going to generate a proxy client to test the Web service.

1. Right-click the wsdl file in the Project explorer view.
2. Select **Web Services** → **Generate Client**.

   This displays the Web services wizard. There is much richer functionality here than just generating a test client. The slide bar allows you to select the stages of Web service client development. The wizard will take you through the stages Develop - Assemble - Deploy - Install - Test.

   We are only interested in testing the Web service using a proxy Java client, so slide the bar to the **Test** position, as shown in Figure 8-5.

   There are also options in the Configuration section of the panel in Figure 8-5 to select the server, runtime, and to change the Web service project name and Web service EAR name that are auto-generated. Because this is just a unit test we left the default of WebServiceProject and WebServiceProjectEAR.



Review your Web service options and make any necessary changes before proceeding to the next page.

Service definition: /redbook/GetHash.wsdl

Client type: Java Proxy

Configuration:
Server: WebSphere v6.1 Server
Web service runtime: IBM WebSphere JAX-RPC
Client project: WebServiceProject
Client EAR project: WebServiceProjectEAR

☐ Monitor the Web service

☑ Overwrite files without warning

☐ Do not show me this dialog box again.

Figure 8-5 Web Services wizard

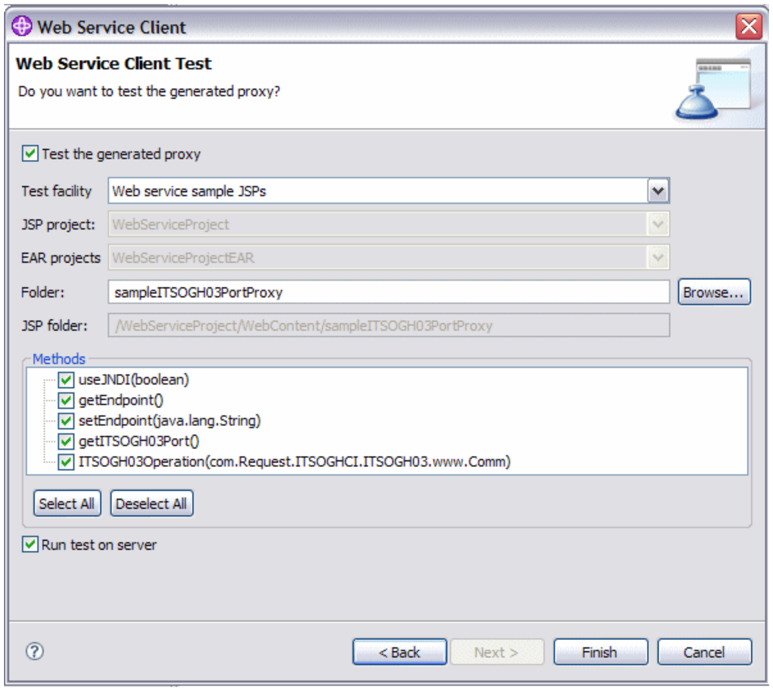
a. Select **Next**.

b. On the Web Service Proxy page, select **Next**.

c. The Web Service Client test page in Figure 8-6 shows all the methods that were generated for the client. Ensure that the following boxes are checked:

- Test the generated proxy
- Run test on server

d. Click **Finish**.



☑ Test the generated proxy

Test facility: Web service sample JSPs

JSP project: WebServiceProject

EAR projects: WebServiceProjectEAR

Folder: sampleITSOGH03PortProxy

JSP folder: /WebServiceProject/WebContent/sampleITSOGH03PortProxy

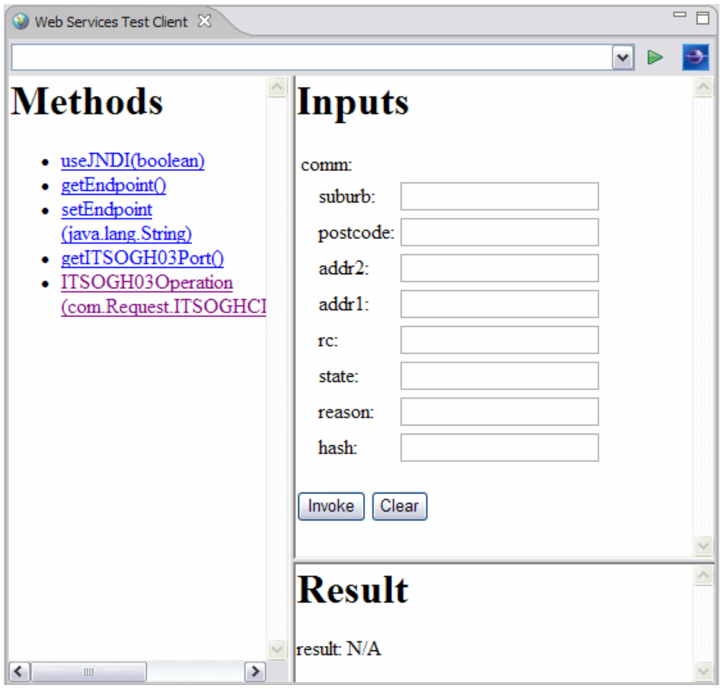
Methods

- ☑ useJNDI(boolean)
- ☑ getEndpoint()
- ☑ setEndpoint(java.lang.String)
- ☑ getITSOGH03Port()
- ☑ ITSOGH03Operation(com.Request.ITSOGHCI.ITSOGH03.www.Comm)

☑ Run test on server

Figure 8-6 Web Service Client Test

3. The Wizard then opens up a Web browser in the work space in which the methods that were created are listed and can be tested. Figure 8-7 on page 194 shows the method that we are testing is ITSOGH03Operation.



Methods

- useJNDI(boolean)
- getEndpoint()
- setEndpoint (java.lang.String)
- getITSOGH03Port()
- ITSOGH03Operation (com.Request.ITSOGHCI

Inputs

comm:

suburb:	
postcode:	
addr2:	
addr1:	
rc:	
state:	
reason:	
hash:	

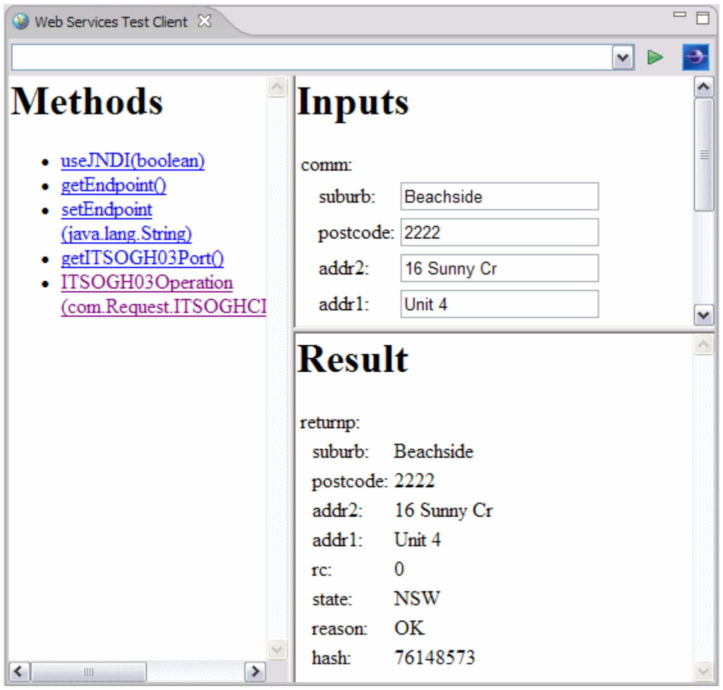
Result

result: N/A

Figure 8-7 Testing the Proxy Client

4. Fill in some values, and click **Invoke**.

   The result is seen in Figure 8-8 on page 195.



Web Services Test Client

Methods

- useJNDI(boolean)
- getEndpoint()
- setEndpoint (java.lang.String)
- getITSOGH03Port()
- ITSOGH03Operation (com.Request.ITSOGHCI

Inputs

comm:

suburb: Beachside

postcode: 2222

addr2: 16 Sunny Cr

addr1: Unit 4

Result

```
returnp:
  suburb:   Beachside
  postcode: 2222
  addr2:    16 Sunny Cr
  addr1:    Unit 4
  rc:       0
  state:    NSW
  reason:   OK
  hash:     76148573
```

Figure 8-8 Successful Test Result

This test was successful!

8.3 A client developed in WebSphere Message Broker

In this section we discuss how we can create a message flow using WebSphere Message Broker V6.0 to perform the client side of our Change of Address application.

WebSphere Message Broker (WMB) is ideally suited to this client task as it is by nature an asynchronous subsystem. WMB allows easy facilitation of the Web service calls via native HTTPRequest nodes and the new ability to import WSDL files as message definition files.

WMB is much too large a topic to be introduced at more than a superficial level here. We briefly introduced WebSphere Message Broker in Chapter 7, “Configuring publication/subscription” on page 159. For further information, refer to the *WebSphere Message Broker Basics*, SG24-7137 IBM Redbooks publication for an introduction to the broker. However, the level of complexity of this message flow is basic, so the reader should have little difficulty following the logic of this approach.

8.3.1 Creating the RetrieveAddressWeb Message Flow

The RetrieveAddress message flow should perform the following tasks:

- Retrieve an address hash from a subscription queue.
- Check the local customer database for any clients whose address matches this address hash.
- Invoke the StandardName Web service for each name returned to obtain the standard format of this entity.
- Call the RetrieveAddress CICS Web service for each matching client, supplying the standard-name of the client and the address hash.
- Update the local customer database with the changed address, if RetrieveAddress returns a new address for client.

The Customer database

The Customer database can be just about any type of data repository required, but in most cases, this will be a database. But it could just as well be a spreadsheet or a VSAM dataset on a mainframe. In this implementation, the database is a DB2 table on Windows. For simplicity, we created a single, linear table to hold our customer data. It looks similar to Figure 8-9 on page 197.

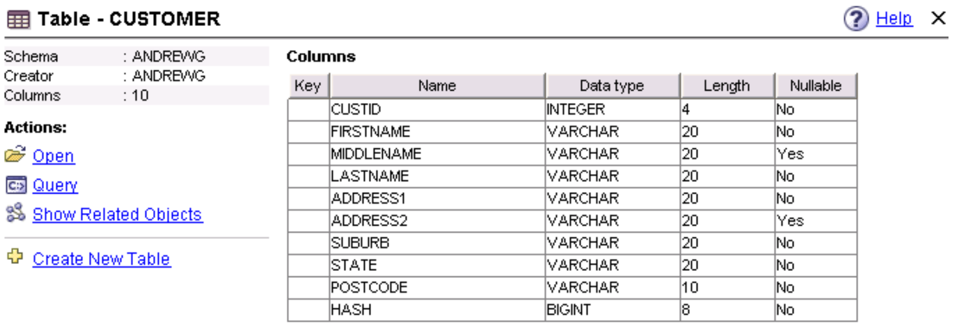


Table - CUSTOMER Help

Schema	: ANDREWG
Creator	: ANDREWG
Columns	: 10

Actions:

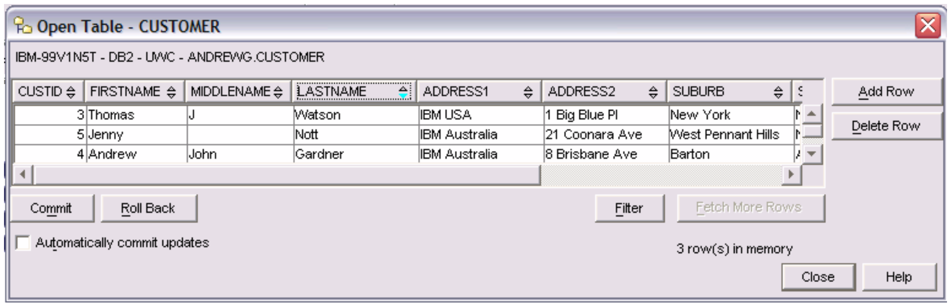
- Open
- Query
- Show Related Objects
- Create New Table

Columns

Key	Name	Data type	Length	Nullable
	CUSTID	INTEGER	4	No
	FIRSTNAME	VARCHAR	20	No
	MIDDLENAME	VARCHAR	20	Yes
	LASTNAME	VARCHAR	20	No
	ADDRESS1	VARCHAR	20	No
	ADDRESS2	VARCHAR	20	Yes
	SUBURB	VARCHAR	20	No
	STATE	VARCHAR	20	No
	POSTCODE	VARCHAR	10	No
	HASH	BIGINT	8	No

Figure 8-9 The Customer database table format

Now since the RetrieveAddress Web service returns no information about the identity of the user for which it publishes a hash, we must already have clients in the Customer database table, together with hash values of their current addresses. This could be done using DB2 Control Center's spreadsheet-like graphical input or some simple SQL INSERTS—which is what is more likely to happen in a real business situation. The Hash code is returned by a call to the GetHash CICS Web service. We created the following entries in our Customer database table with these hash codes (not shown).


Open Table - CUSTOMER

IBM-99V1N5T - DB2 - UWC - ANDREWG.CUSTOMER

CUSTID	FIRSTNAME	MIDDLENAME	LASTNAME	ADDRESS1	ADDRESS2	SUBURB	S
3	Thomas	J	Watson	IBM USA	1 Big Blue Pl	New York	N
5	Jenny		Nott	IBM Australia	21 Coonara Ave	West Pennant Hills	N
4	Andrew	John	Gardner	IBM Australia	8 Brisbane Ave	Barton	A

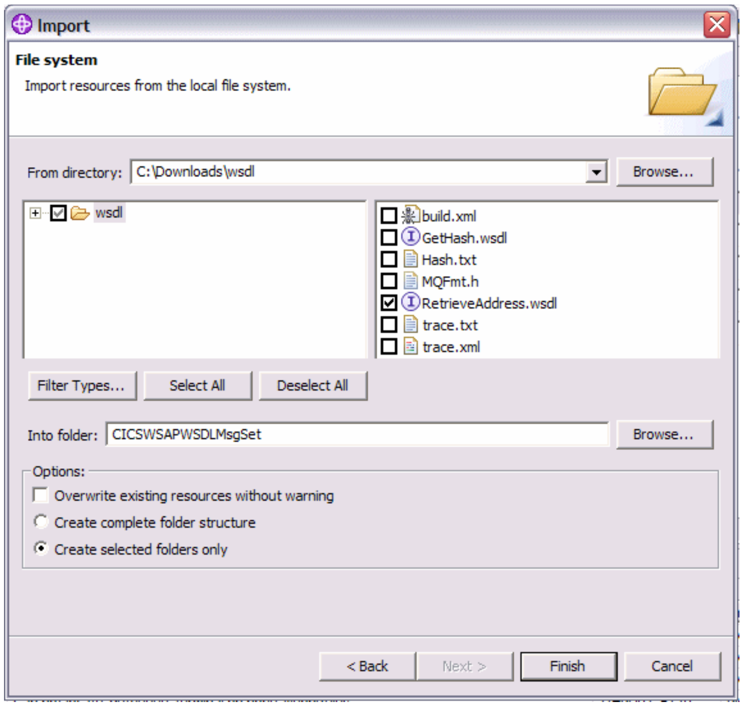
Figure 8-10 Sample contents of the Customer database table

Creating a Message Set from the RetrieveAddress WSDL

Just as we created a message set in the broker by importing a C Header structure, we now create a message set for the call to the RetrieveAddress message set.

1. Create a new Message Set project. As previously mentioned, it is best practice to separate different classes of messages. We call both the Message Set project and Message Set CICSWSAPWSDLMsgSet. The physical format will be XML because WSDL is an XML schema.

2. Import the WSDL file into the work space before creating a Message Definition file from it. (File → Import → File System.)
3. Locate the RetrieveAddress.wsdl file and import it into the CICSWSAPWSDLMsgSet project.


Import

File system

Import resources from the local file system.

From directory: C:\Downloads\wsdl

[✓] wsdl

- [ ] build.xml
- [ ] GetHash.wsdl
- [ ] Hash.txt
- [ ] MQFmt.h
- [✓] RetrieveAddress.wsdl
- [ ] trace.txt
- [ ] trace.xml

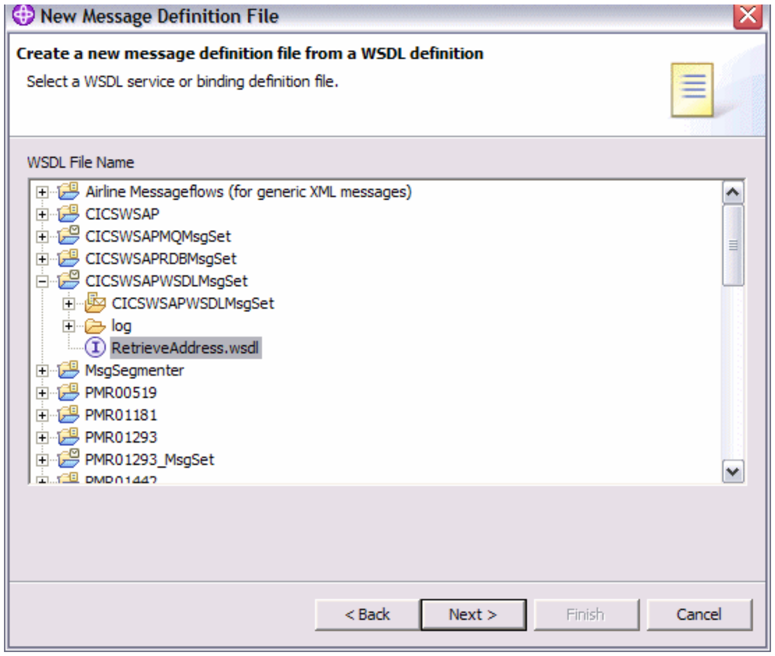
Into folder: CICSWSAPWSDLMsgSet

Options:

- [ ] Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

Figure 8-11 Import the RetrieveAddress WSDL file

4. Create the Message Definition, by selecting **File** → **New** → **Message Definition File**, and then choose**: WSDL File**.
5. Locate the WSDL file in the CICSWSAPWSDLMsgSet project.


New Message Definition File

Create a new message definition file from a WSDL definition

Select a WSDL service or binding definition file.

WSDL File Name

- Airline Messageflows (for generic XML messages)
- CICSWSAP
- CICSWSAPMQMsgSet
- CICSWSAPRDBMsgSet
- CICSWSAPWSDLMsgSet
  - CICSWSAPWSDLMsgSet
  - log
  - RetrieveAddress.wsdl
- MsgSegmenter
- PMR00519
- PMR01181
- PMR01293
- PMR01293_MsgSet
- PMR01442

Figure 8-12 Locate the WSDL file

6. Click **Next**. We are asked what Message set in which we want to save the definition. Choose CICSWSAPWSDLMsgSet, and click Next. We take the default on WSDL Warnings and leave the NameSpaces as is. Take defaults until the message definition file is created.

Constructing the RetrieveAddress Message Flow

Our RetrieveAddress message flow looks similar to Figure 8-13.

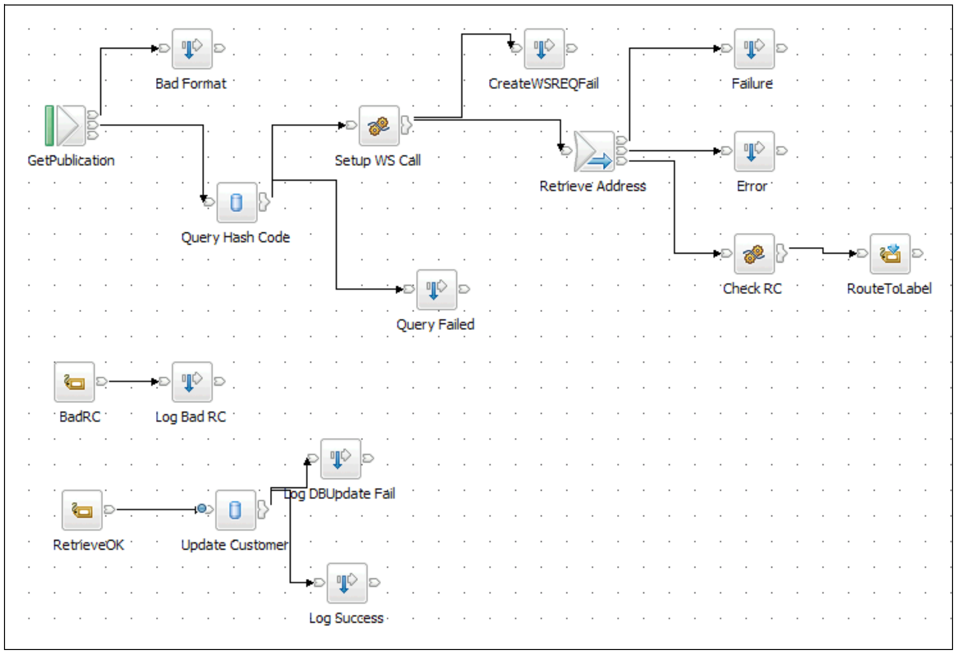
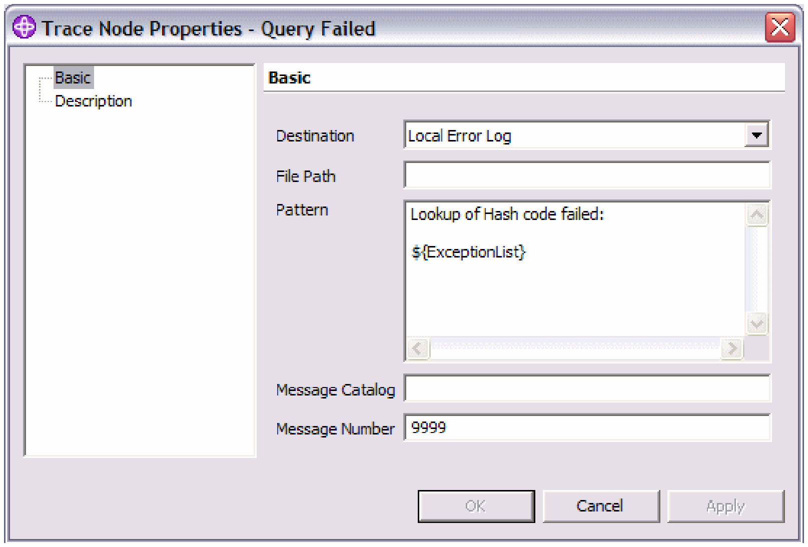


Figure 8-13 The RetrieveAddress message flow

Most of the nodes in this message flow are **Trace** nodes, which are essentially there to help with run-time errors. They are all configured to write to the *Local Error Log*. For example the *Query Failed* node looks similar to Figure 8-14 on page 201.


Trace Node Properties - Query Failed

Basic

Destination: Local Error Log

File Path:

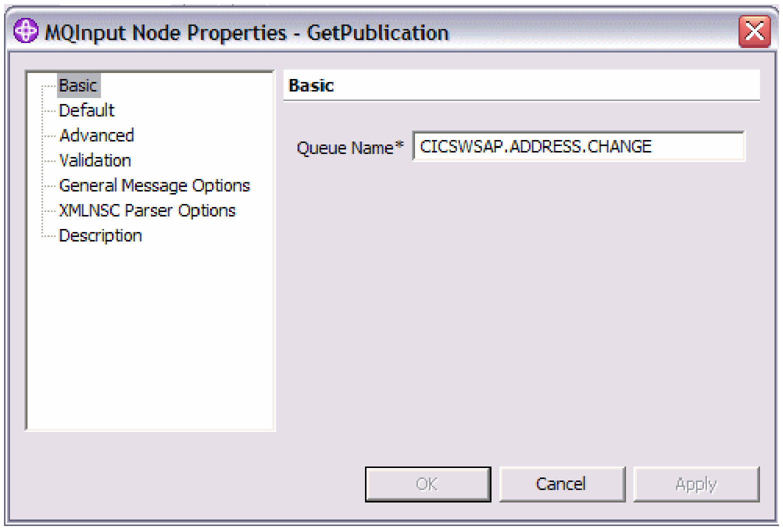
Pattern:
```
Lookup of Hash code failed:

${ExceptionList}
```

Message Catalog:

Message Number: 9999

Figure 8-14 A typical Trace node

Ignoring error conditions for now, the Message Flow commences with the receipt of a published hash onto our subscription queue. Our queue name was indicated on our subscription request and matches the value on the MQInput node.

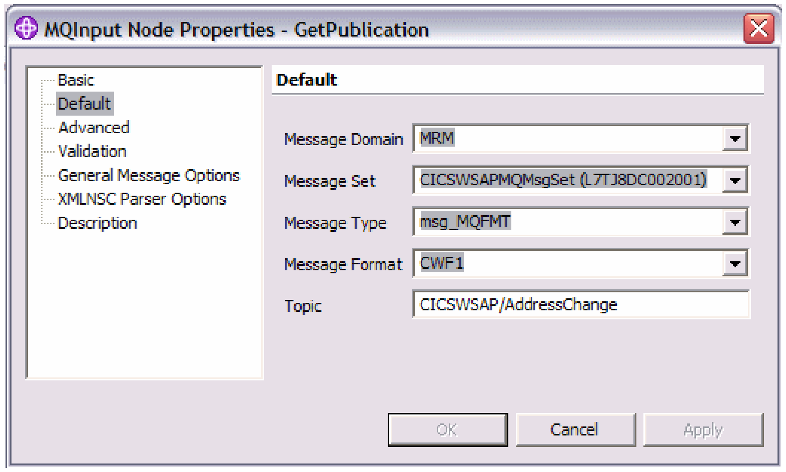

MQInput Node Properties - GetPublication

Basic

Queue Name*: CICSWSAP.ADDRESS.CHANGE

Figure 8-15 The subscription queue on the MQInput node

Essential information about the structure of the message is indicated in the *Default* tab.



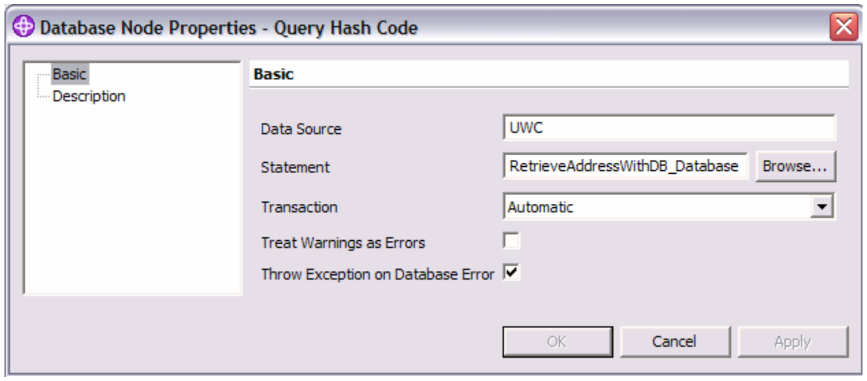
MQInput Node Properties - GetPublication

Default

Message Domain	MRM
Message Set	CICSWSAPMQMsgSet (L7TJ8DC002001)
Message Type	msg_MQFMT
Message Format	CWF1
Topic	CICSWSAP/AddressChange

Figure 8-16 The Message format data

So, at this stage, the MQInput node knows what queue it takes messages from and also understands the format of the message. The parsed message tree is passed to the next node, which is a query to our local Customer database table to see if we have any clients with a matching hash key. The Database node needs to know what database it is using.



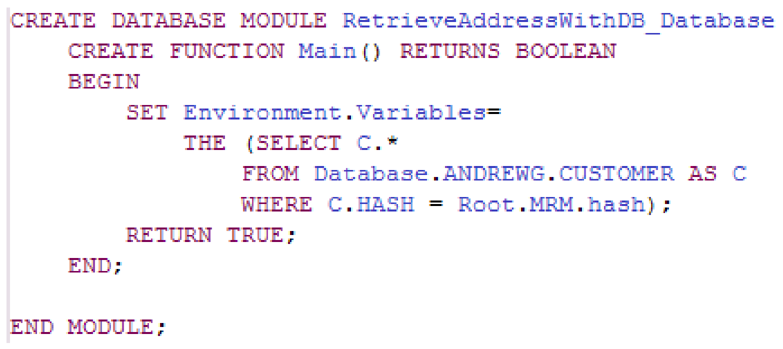
Database Node Properties - Query Hash Code

Basic

Data Source	UWC
Statement	RetrieveAddressWithDB_Database
Transaction	Automatic
Treat Warnings as Errors	☐
Throw Exception on Database Error	✓

Figure 8-17 The Database Node properties

In this case, our database is called UWC (Unified Widget Corporation), which contains just one user table called **Customer**. The **Statement** field indicates the name of an ESQL module containing the database query, which is follows in Figure 8-18 on page 203.

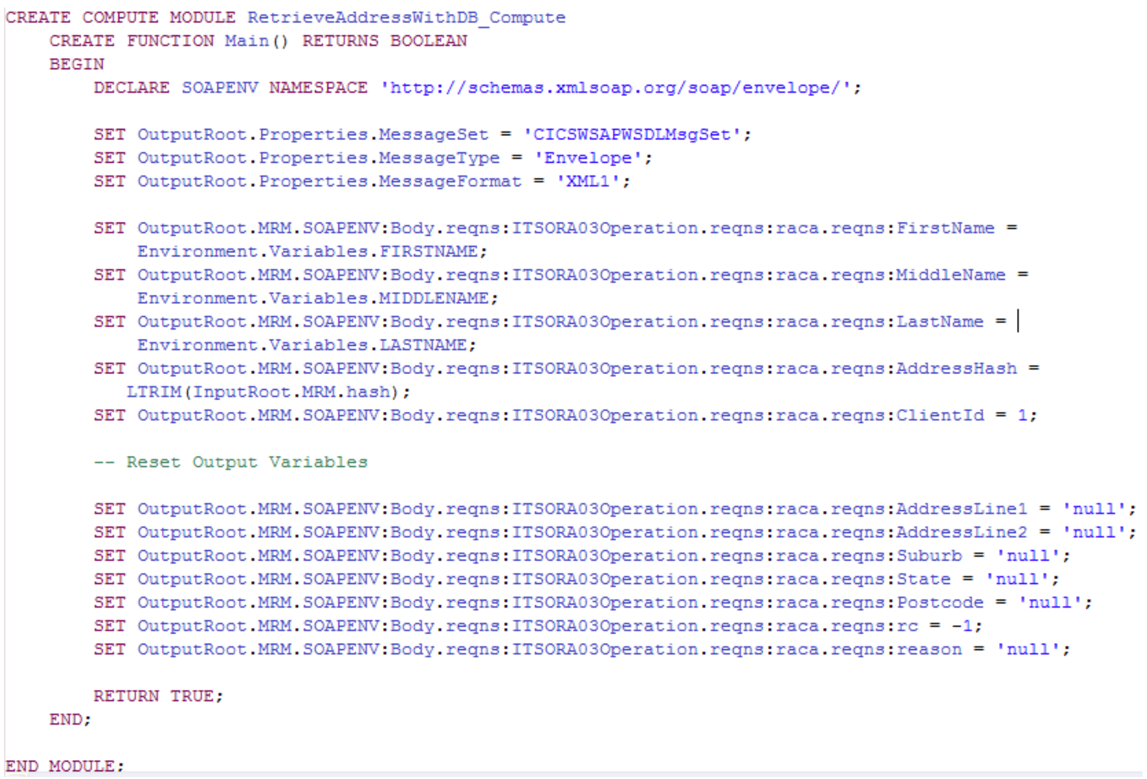


```
CREATE DATABASE MODULE RetrieveAddressWithDB_Database
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        SET Environment.Variables=
            THE (SELECT C.*
                FROM Database.ANDREWG.CUSTOMER AS C
                WHERE C.HASH = Root.MRM.hash);
        RETURN TRUE;
    END;

END MODULE;
```

Figure 8-18 The query

This ESQL places the result of the query in the Environment *branch* of the parse tree where this result will be available to subsequent nodes.

The next significant node in the message flow is a Compute node, which constructs the call to the RetrieveAddress Web service. It takes the results of the above ESQL query and places the first, middle, and last names, as well as the hash value, into the Web service call parameter structure.

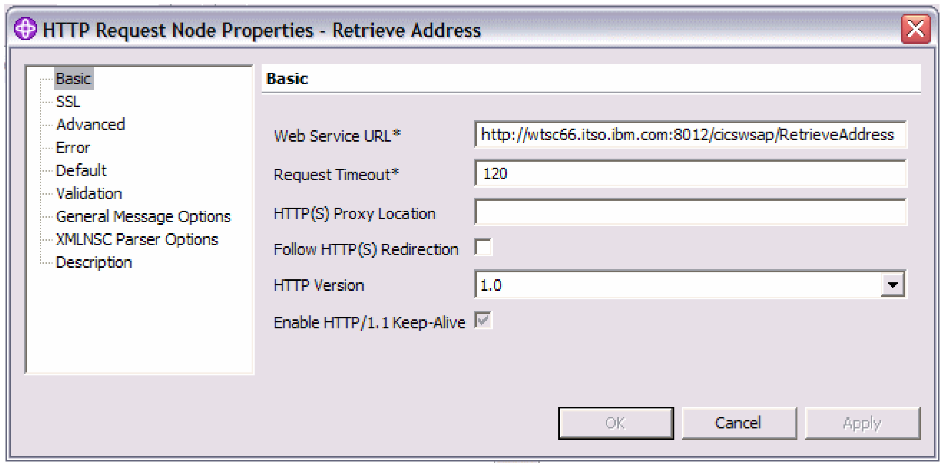


```
CREATE COMPUTE MODULE RetrieveAddressWithDB_Compute
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        DECLARE SOAPENV NAMESPACE 'http://schemas.xmlsoap.org/soap/envelope/';

        SET OutputRoot.Properties.MessageSet = 'CICSWSAPWSDLMsgSet';
        SET OutputRoot.Properties.MessageType = 'Envelope';
        SET OutputRoot.Properties.MessageFormat = 'XML1';

        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:FirstName =
            Environment.Variables.FIRSTNAME;
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:MiddleName =
            Environment.Variables.MIDDLENAME;
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:LastName =
            Environment.Variables.LASTNAME;
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressHash =
           LTRIM(InputRoot.MRM.hash);
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:ClientId = 1;

        -- Reset Output Variables

        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine1 = 'null';
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine2 = 'null';
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Suburb = 'null';
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:State = 'null';
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Postcode = 'null';
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:rc = -1;
        SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:reason = 'null';

        RETURN TRUE;
    END;

END MODULE;
```

Figure 8-19 Setting up the call to the RetrieveAddress Web service

The output of this node is now suitable for the following node, the actual call to the Web service via the HTTPRequest node. Essential information about this node includes the following:

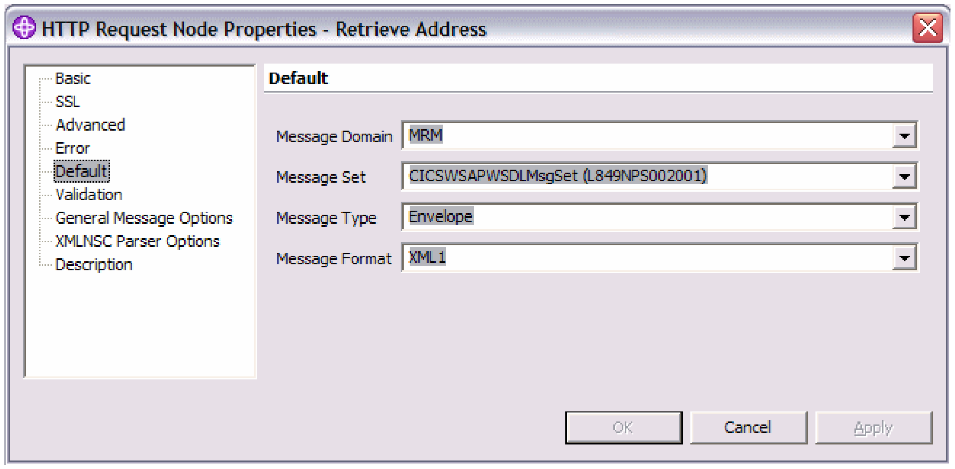
- The actual URL of the Web service
- Transport properties like whether SSL is in use
- The names of the message sets, so we can interpret the results of the Web service call and parse the results ready for the next node


HTTP Request Node Properties - Retrieve Address

Basic

Web Service URL* http://wtsc66.itso.ibm.com:8012/cicswsap/RetrieveAddress
Request Timeout* 120
HTTP(S) Proxy Location
Follow HTTP(S) Redirection ☐
HTTP Version 1.0
Enable HTTP/1.1 Keep-Alive ✓

Figure 8-20 The HTTPRequest node properties - 1

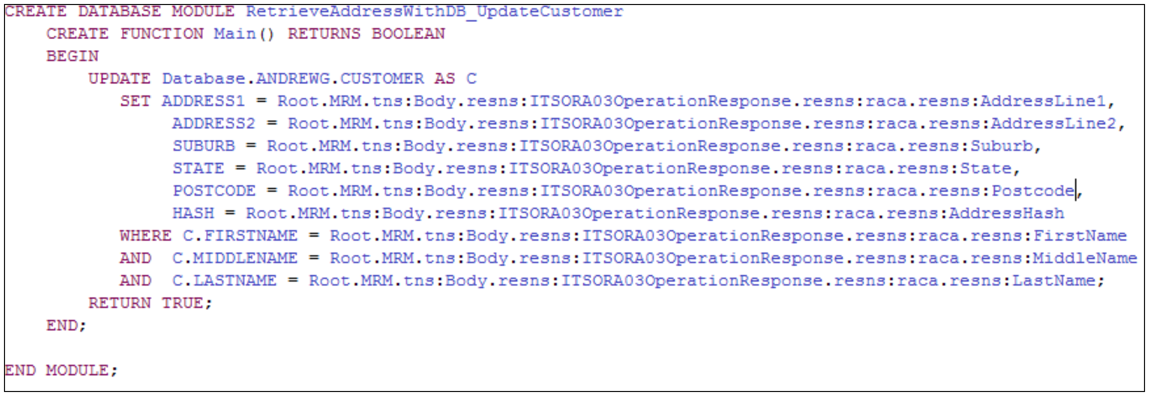

HTTP Request Node Properties - Retrieve Address

Default

Message Domain MRM
Message Set CICSWSAPWSDLMsgSet (L849NPS002001)
Message Type Envelope
Message Format XML1

Figure 8-21 The HTTPRequest node properties - 2

The next Compute node in the sequence simply checks the success or otherwise of the Web service call and sets an indicator for the next node: the RouteToLabel, depending on the error condition. If successful, the next significant node is the UpdateCustomer Database node. Here, we attempt to update the client record in the local DB2 table with the results of the RetrieveAddress Web service call. The properties of this node indicate the database being targeted: UWC. This node contains ESQL to perform the update.



```
CREATE DATABASE MODULE RetrieveAddressWithDB_UpdateCustomer
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        UPDATE Database.ANDREWG.CUSTOMER AS C
           SET ADDRESS1 = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressLine1,
                ADDRESS2 = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressLine2,
                SUBURB = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:Suburb,
                STATE = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:State,
                POSTCODE = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:Postcode,
                HASH = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressHash
           WHERE C.FIRSTNAME = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:FirstName
           AND  C.MIDDLENAME = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:MiddleName
           AND  C.LASTNAME = Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:LastName;
        RETURN TRUE;
    END;

END MODULE;
```

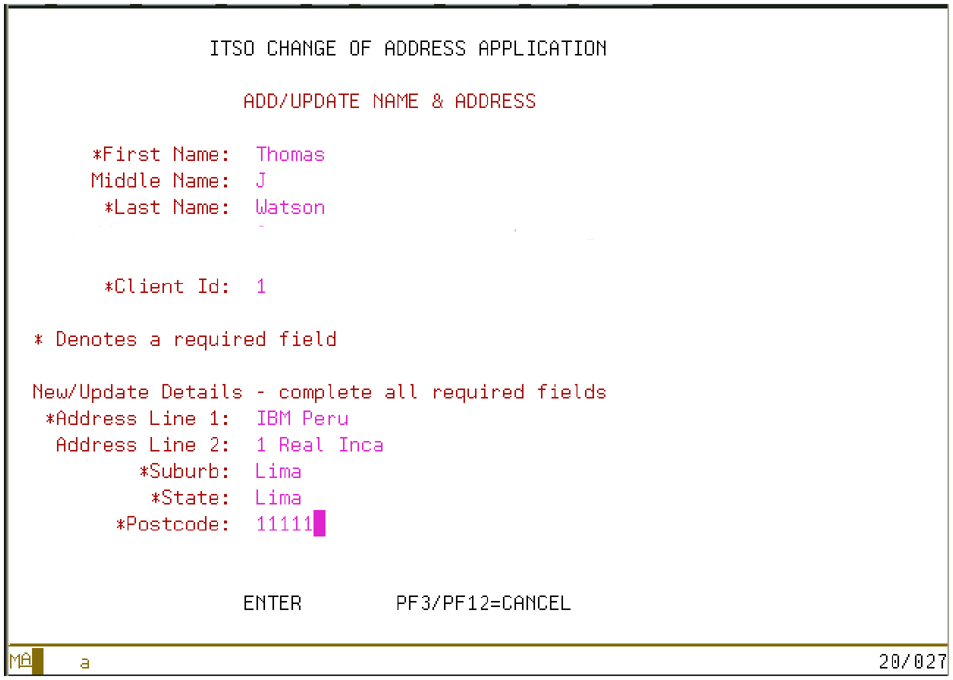
Figure 8-22 Updating the local Customer database

And with the success of this update, the message flow fulfilled it's purpose.

Testing the RetrieveAddress Message Flow

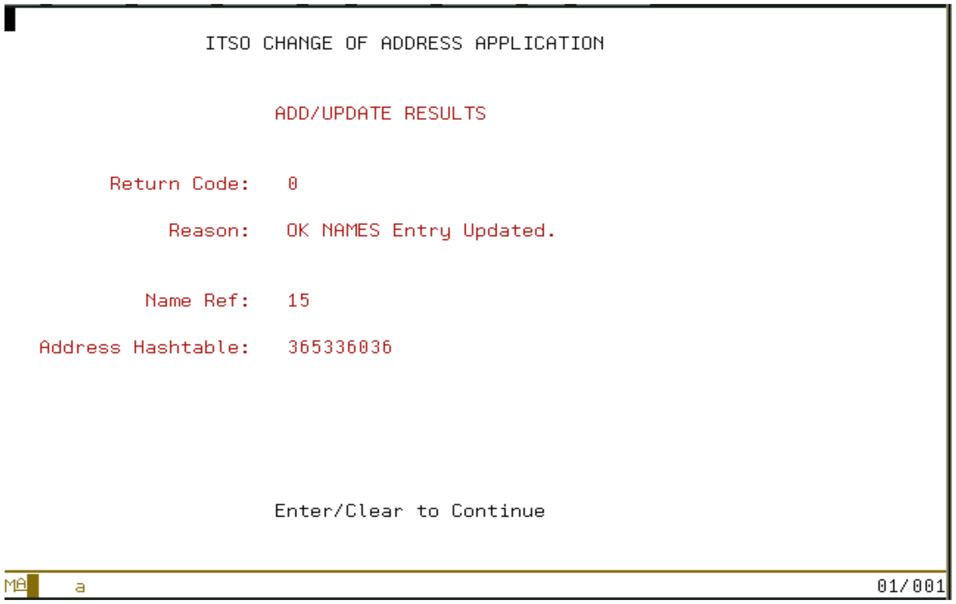
We can now perform an end-to-end test of the AddressChange message flow presuming everything described so far is in place. It should now just be a case of updating one of the known customer's address details on the CICS application (Option 3). This generates the hash, which is published by our Hash Publication message flow. All registered subscribers, including our RetrieveAddress message flow receive this message. If the message flow runs through to successful completion, we expect to see a message logged indicating the success, as well as an update to our Customer database table.

1. First, we update the address of Thomas J. Watson.



```
                 ITSO CHANGE OF ADDRESS APPLICATION

                    ADD/UPDATE NAME & ADDRESS

        *First Name:  Thomas
        Middle Name:  J
         *Last Name:  Watson


         *Client Id:  1

 * Denotes a required field

 New/Update Details - complete all required fields
  *Address Line 1:  IBM Peru
   Address Line 2:  1 Real Inca
          *Suburb:  Lima
           *State:  Lima
        *Postcode:  11111


                   ENTER         PF3/PF12=CANCEL

MA    a                                                   20/027
```

Figure 8-23 Change Thomas J Watson's address



```
                 ITSO CHANGE OF ADDRESS APPLICATION


                       ADD/UPDATE RESULTS


        Return Code:    0

             Reason:    OK NAMES Entry Updated.


           Name Ref:    15

 Address Hashtable:    365336036




                     Enter/Clear to Continue

MA    a                                                    01/001
```

Figure 8-24 Update confirmed

2. Look at the Windows Event Viewer for evidence of the success or otherwise of the update on the client side.

Example 8-2 An error log event indicating success of the update

```
The description for Event ID ( 19003 ) in Source ( WebSphere Broker v6002 ) could not be found.
It contains the following insertion string(s): .
AJGBROKER1.WSTest
'
Success !!

(
  (0x01000000):Properties        = (
    (0x03000000):MessageSet      = 'L849NPS002001'
    (0x03000000):MessageType     = 'Envelope'
    (0x03000000):MessageFormat   = 'XML1'
    (0x03000000):Encoding        = 546
    (0x03000000):CodedCharSetId  = 1208
    (0x03000000):Transactional   = FALSE
    (0x03000000):Persistence     = FALSE
    (0x03000000):CreationTime    = GMTTIMESTAMP '2007-02-14 09:42:03.607'
    (0x03000000):ExpirationTime  = -1
    (0x03000000):Priority        = 0
    (0x03000000):ReplyIdentifier = X'000000000000000000000000000000000000000000000000'
    (0x03000000):ReplyProtocol   = 'MQ'
    (0x03000000):Topic           = NULL
    (0x03000000):ContentType     = 'text/xml; charset="UTF-8"'
  )
  (0x01000000):HTTPResponseHeader = (
    (0x03000000):X-Original-HTTP-Status-Line = 'HTTP/1.0 200 OK'
    (0x03000000):X-Original-HTTP-Status-Code = 200
    (0x03000000):Server                      = 'IBM_CICS_Transaction_Server/3.1.0(zOS)'
    (0x03000000):Date                        = 'Thu, 15 Feb 2007 00:21:05 GMT'
    (0x03000000):Content-Length              = '00001293'
    (0x03000000):Content-Type                = 'text/xml; charset="UTF-8"'
  )
  (0x0100001B):MRM               = (
    (0x01000013)http://schemas.xmlsoap.org/soap/envelope/:Body = (
      (0x0100001B)http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationResponse = (
        (0x01000013)http://www.ITSORA03.ITSORACA.Response.com:raca = (
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:FirstName    = 'Thomas
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:MiddleName   = 'J '
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:LastName     = 'Watson '
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:AddressLine1 = 'IBM Peru '
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:AddressLine2 = '1 Real Inca '
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:Suburb       = 'Lima '
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:State        = 'Lima      '
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:Postcode     = '11111     '
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:AddressHash  = 365336036
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:ClientId     = 1
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:NameRef      = 0
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:rc           = 0
          (0x0300000B)http://www.ITSORA03.ITSORACA.Response.com:reason       = 'OK, new address
details supplied'
        )
      )
    )
  )
)
'
RetrieveAddressWithDB.Log Success
```

3. Ensure that the Customer database table update occurred. A simple SQL query will suffice, similar to Example 8-3:

Example 8-3 Querying the CUSTOMER database

```
db2 => select * from CUSTOMER where LASTNAME='Watson'

CUSTID               FIRSTNAME            MIDDLENAME           LASTNAME
    ADDRESS1             ADDRESS2             SUBURB
STATE
        POSTCODE   HASH
-------------------- -------------------- --------------------
-----------------
--- -------------------- -------------------- --------------------
-------------
------- ---------- --------------------
                   1 Thomas               J                    Watson
    IBM Peru             1 Real Inca          Lima                 Lima
        11111                365336036

  1 record(s) selected.
```

The message flow performed its designated task.

Running the message flow on a System z Broker

Although this message flow was developed on Windows, we tested it on a System z WebSphere Message Broker. We chose to run a cut-down version of the message flow, leaving out the database nodes. This was purely due to time constraints.

We discovered a couple of changes were necessary. When we first tried to run the message flow, we could see no trace node output. Clearly *Local Error Log* is not appropriate as a destination for System z broker trace nodes. We changed this to *File* with /var/wmqi/MQ8GBRK/output/RetrieveAddress.log as the destination.

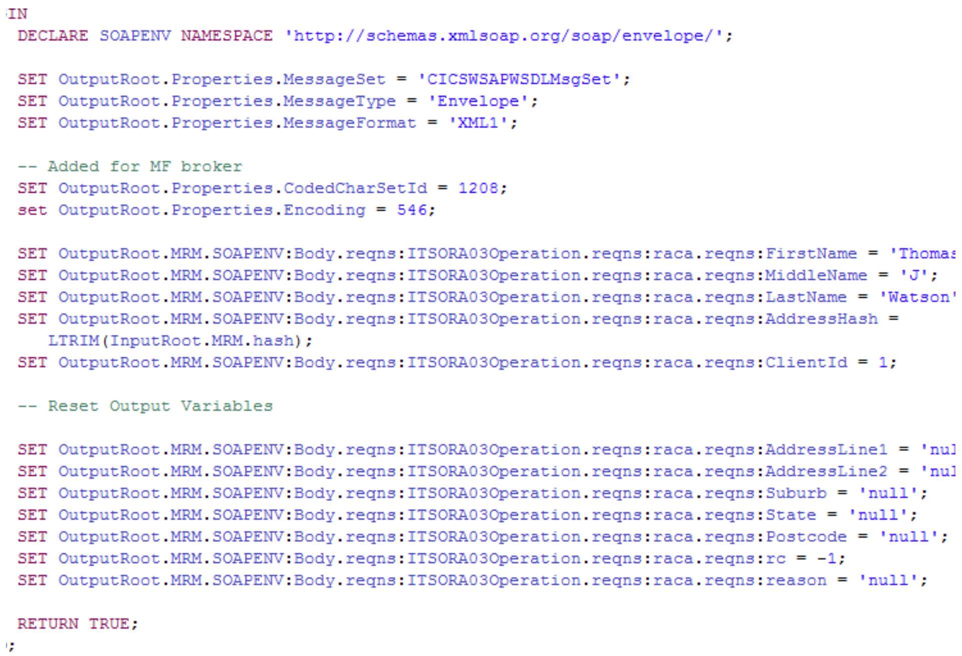
Secondly, out RetrieveAddress Web service call failed with the diagnostic in Example 8-4:

Example 8-4 Error in RetrieveAddress Web service call

```
Error during RetrieveAddress WS call:.
.
(
  (0x01000000):Properties         = (
    (0x03000000):MessageSet      = 'CICSWSAPWSDLMsgSet (L849NPS002001)'
    (0x03000000):MessageType     = 'Envelope'
    (0x03000000):MessageFormat   = 'XML1'
    (0x03000000):Encoding        = 546
    (0x03000000):CodedCharSetId  = 1208
    (0x03000000):Transactional   = FALSE
    (0x03000000):Persistence     = FALSE
    (0x03000000):CreationTime    = GMTTIMESTAMP '2007-02-17 05:00:04.135811'
    (0x03000000):ExpirationTime  = -1
    (0x03000000):Priority        = 0
    (0x03000000):ReplyIdentifier =X'000000000000000000000000000000000000000000000000'
    (0x03000000):ReplyProtocol   = 'MQ'
    (0x03000000):Topic           = NULL
    (0x03000000):ContentType     = 'text/xml; charset="UTF-8"'
  )
  (0x01000000):HTTPResponseHeader = (
    (0x03000000):X-Original-HTTP-Status-Line = 'HTTP/1.0 500 Internal Server
Error'
    (0x03000000):X-Original-HTTP-Status-Code = 500
    (0x03000000):Server = 'IBM_CICS_Transaction_Server/3.1.
    (0x03000000):Date                       = 'Sat, 17 Feb 2007 05:02:12 GMT'
    (0x03000000):Content-Length             = '00000516'
    (0x03000000):Content-Type               = 'text/xml; charset="UTF-8"'
  )
  (0x01000000):BLOB               = (
    (0x03000000):UnknownParserName = ''
    (0x03000000):BLOB              =
X'3c534f41502d454e563a456e76656c6f706520786....
  )
)
```

We reasoned that the only difference between the Windows and System z platform was the code-page and encoding schemes. We added two lines at the start of the our ESQL on the *Setup WS Call* compute node to set the CodedCharacterSet and Encoding values as *suggested* by the trace output above.

This change worked as we wanted. Figure 8-25 shows the modified ESQL:



```
IN
 DECLARE SOAPENV NAMESPACE 'http://schemas.xmlsoap.org/soap/envelope/';

 SET OutputRoot.Properties.MessageSet = 'CICSWSAPWSDLMsgSet';
 SET OutputRoot.Properties.MessageType = 'Envelope';
 SET OutputRoot.Properties.MessageFormat = 'XML1';

 -- Added for MF broker
 SET OutputRoot.Properties.CodedCharSetId = 1208;
 set OutputRoot.Properties.Encoding = 546;

 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:FirstName = 'Thomas
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:MiddleName = 'J';
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:LastName = 'Watson'
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressHash =
    LTRIM(InputRoot.MRM.hash);
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:ClientId = 1;

 -- Reset Output Variables

 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine1 = 'nul
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine2 = 'nul
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Suburb = 'null';
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:State = 'null';
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Postcode = 'null';
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:rc = -1;
 SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:reason = 'null';

 RETURN TRUE;
;
```

Figure 8-25 The modified ESQL

Possible improvements

- Usually more than one person lives at an address. This sample message flow needs to be able to handle multiple clients with same address hash.
- Use HTTPS on the RetrieveAddress Web service call to secure possibly personal information

- Message flow should invoke the StandardName Web service before the RetrieveAddress Web service. These are currently dummy functions. See the discussion on StandardName and StandardAddress in Chapter 5.

9

Tracing the Change of Address scenario

In this chapter we take a low-level look at the processes and interactions of the complete Change of Address scenario. We take simultaneous, detailed traces in CICS and both the System z broker and the Windows broker. These traces are interleaved to show the processing happening in each component as a time sequence with annotations.

We include detailed instructions for collecting this type of trace information.

9.1 Collecting the traces

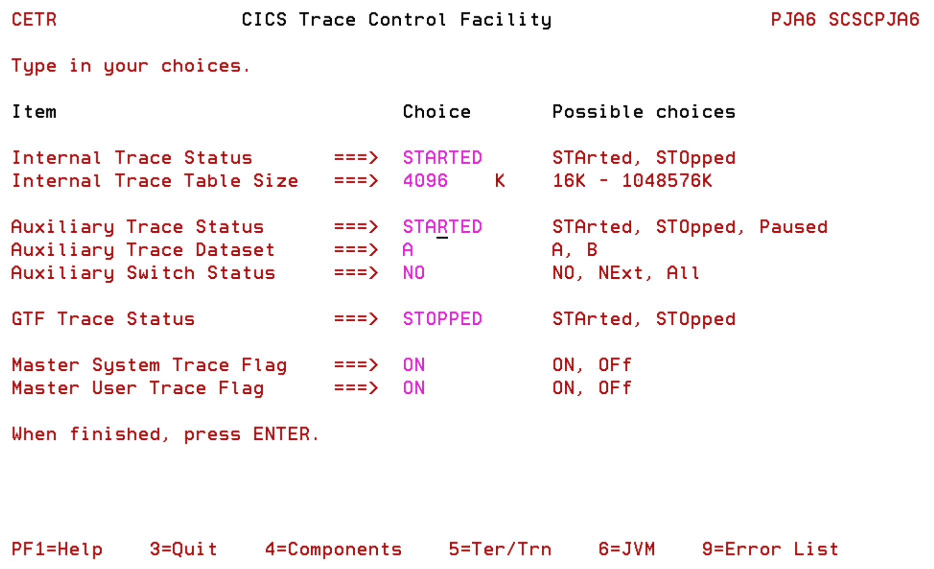
The Change of Address scenario is probably the most complex of the various application scenarios as it involves at least three components across two platforms:

1. **CICS on System z**
   - Acts as the service provider
   - Runs the ‘back end’ program
   - Interacts with DB2 on System z
   - Writes a message to a WebSphere MQ queue
2. **WebSphere Message Broker on System z**
   - A message flow detects a message from CICS
   - WMB publishes this message to subscriber message queues
3. **WebSphere Message Broker on Windows**
   - A message flow detects a message from WMB on its subscription queue
   - Initiates a Web service call in CICS for retrieval of information

We say at least three components because we choose not to trace WebSphere MQ flows, as these are fairly straightforward and well indicated in the WebSphere Message Broker and CICS traces. We recommend tracing WMQ if suspected problems with messaging existed.

9.1.1 Tracing the Web service on CICS

The tool of choice for tracing CICS transactions is the Auxiliary trace. The *auxtrace* is highly configurable and simple to use. To start and configure the CICS auxtrace, use the CICS Trace Control Facility. This is provided with the CETR transaction. Figure 9-1 on page 215 shows the options we used to start the trace.



```
CETR                 CICS Trace Control Facility                   PJA6 SCSCPJA6

Type in your choices.

Item                               Choice       Possible choices

Internal Trace Status        ===>  STARTED      STArted, STOpped
Internal Trace Table Size    ===>  4096    K    16K - 1048576K

Auxiliary Trace Status       ===>  STARTED      STArted, STOpped, Paused
Auxiliary Trace Dataset      ===>  A            A, B
Auxiliary Switch Status      ===>  NO           NO, NExt, All

GTF Trace Status             ===>  STOPPED      STArted, STOpped

Master System Trace Flag     ===>  ON           ON, OFf
Master User Trace Flag       ===>  ON           ON, OFf

When finished, press ENTER.



PF1=Help    3=Quit    4=Components    5=Ter/Trn    6=JVM    9=Error List
```

Figure 9-1 CETR Transaction

The relevant options in CETR are as follows:

Auxiliary Trace Status - Over-type the choice to start or stop the trace.

> **Tip:** As CICS can generate masses of trace data, only Start the trace when you are ready to begin the transaction being traced. Likewise, Stop the trace as soon as practical after the transaction or operation has finished.

Auxiliary Trace Dataset - here the dataset is specified. There are two auxiliary trace datasets that CICS uses. They are defined and created using the job DEFTRCDS with the DD names DFHAUXT for the 'A' dataset and DFHBUXT for the 'B' dataset.

Auxiliary Switch Status - specifies whether CICS should switch to the other dataset when the current becomes full.

You can filter certain CICS component trace points. As we are interested in CICS Web services, we chose to specify special tracing for the CICS Domains: EI (Exec Interface), WB (Web) and PI (Pipeline Manager). Use the following steps to achieve this:

1. In CETR, select **PF4=Components**.

2. Specify a trace level of two for each of the components EI, WB and PI by over-typing the ‘1’ in the Standard column with ‘2’.
3. Press **ENTER=Change** to save, and then **PF3=Quit**.

Formatting the Trace

After the tracing stops, use the DFHTU640 utility to extract all or selected trace entries and to format the data. We used the job in Example 9-1 to format our data. We use the **TYPETR** option to extract trace entries that we are interested in, and we also specify **ABBREV** to extract abbreviated, one line for each trace entries.

Example 9-1 The Trace Utility program DFHTU640

```
//ITSOAUXT   JOB  ,CCOMP,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,REGION=4M
//*
//* CICS AUX TRACE
//*
//PRINT    EXEC PGM=DFHTU640
//STEPLIB  DD   DSN=CICSTS31.CICS.SDFHLOAD,DISP=SHR
//DFHAUXT  DD   DSN=CICSSYSF.CICS640.PJA6.DFHAUXT,DISP=SHR
//DFHAXPRT DD   SYSOUT=*
//DFHAXPRM DD   *
ABBREV
TYPETR=(DS0000-FFFF,AP0000-FFFF,XM0000-FFFF,WB0000-FFFF,PI0000-FFFF)
```

9.1.2 Tracing the Message Broker flow on distributed platforms

Collecting a user trace for the WebSphere Message Broker on distributed platforms including Windows follows this set of commands:

1. To start the trace:
   mqsichangetrace <Broker-Name> -**u** **-e** <EG-name> **-l** <level>
2. To stop the trace:
   mqsichangetrace <Broker-Name> **-u** **-e** <EG-name> **-l none**
3. To extract the trace data:
   mqsireadlog <Broker-name> **-u** **-e** <EG-name> **-o** <trace-data.xml>
4. To format the trace data:
   mqsiformatlog **-i** <trace-data.xml> **-o** <trace-data.txt>

For this particular trace we use the *User Trace* menu option in the broker toolkit to set the broker trace levels. Do this from the *Broker Administration* perspective in the *Domains* pane.

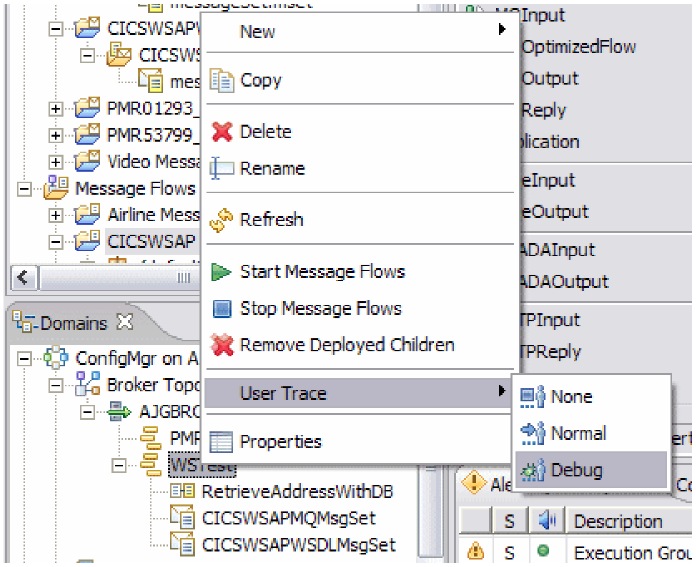


Figure 9-2 Setting the trace level in the broker toolkit

This is equivalent to setting the trace level using the following command:
mqsichangetrace AJGBROKER1 -u -e WSTest -l debug

We stopped the trace using the broker toolkit and ran the following commands to extract and format the trace:

```
mqsireadlog AJGBROKER1 -u -e WSTest -o WSTest.xml

mqsiformatlog -i WSTest.xml -o WSTest.txt
```

9.1.3 Tracing the Message Broker flow on System z

Running traces on a System z broker is similar to the previous section. We have the option of using the toolkit as above or using an MVS operator command:

```
f mq8gbrk,ct u=yes,e=’WSTest’,l=debug
```

However, we found entering the command using SDSF resulted in the WSTest being converted to upper case, resulting in an error due to undefined execution group.

We found that the sequence of steps documented above for collecting a trace on distributed platforms (including Windows) can be run unchanged in a batch UNIX System Services job. For example to start the trace we submitted the JCL in Example 9-2 on page 218.

Example 9-2 Broker Commands Batch job

```
//ITSOBIP JOB ,'WMB COMMAND',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//*****************************************************************
//* Copy ENVFILE to SYSOUT
//*****************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='/u/mq8gbrk/ENVFILE'
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//*****************************************************************
//* Run mqsi command
//*****************************************************************
//*
//BIPLIST  EXEC PGM=IKJEFT01,REGION=0M
//*        DB2 Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=DB8M8.SDSNEXIT
//         DD DISP=SHR,DSN=DB8M8.SDSNLOAD
//         DD DISP=SHR,DSN=DB8M8.SDSNLOD2
//*        MQSeries Runtime Libraries
//         DD DISP=SHR,DSN=MQ600.SCSQANLE
//         DD DISP=SHR,DSN=MQ600.SCSQAUTH
//         DD DISP=SHR,DSN=MQ600.SCSQLOAD
//STDENV   DD PATHOPTS=(ORDONLY),
//            PATH='/u/mq8gbrk/ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
BPXBATSL PGM -
  /usr/lpp/mqsi/mb8g/bin/-
mqsichangetrace MQ8GBRK -u -e WSTest -l debug
/*
//
```

9.2 The annotated trace of the scenario

The following traces describe the most common scenario of the Change of Address application, that is, the actual change of address. In this scenario, the postal service operator makes a change to a *registered* person's address using option 3 of the Change of Address application. This makes the change to the mainframe DB2 table and puts the hash code of the **old** address onto the publication queue. The CICS program finally sends a response screen to the operator indicating success and the new hashcode.

The HashPublication message flow running on the System z broker now publishes the hashcode to all the registered subscribers of the topic: CICSWSAP/AddressChange, of which there are 3 in this trace example. This is the end of the HashPublication message flow.

The third component is a trace of a broker message flow, which represents the client side of the application. The client code receives the notification of the address change via the hashcode publication and checks to see if they have a client with that particular hashvalue. If so, it calls the CICS RetrieveAddress Web service to retrieve the new address for that person. If successful, the message flow updates the local customer database with the new address. This is the end of the sequence.

The flowcharts in the following pages may help to understand the time sequence and interactions of the various components. Note these flowcharts only represent this particular scenario and not the full logic of the application.

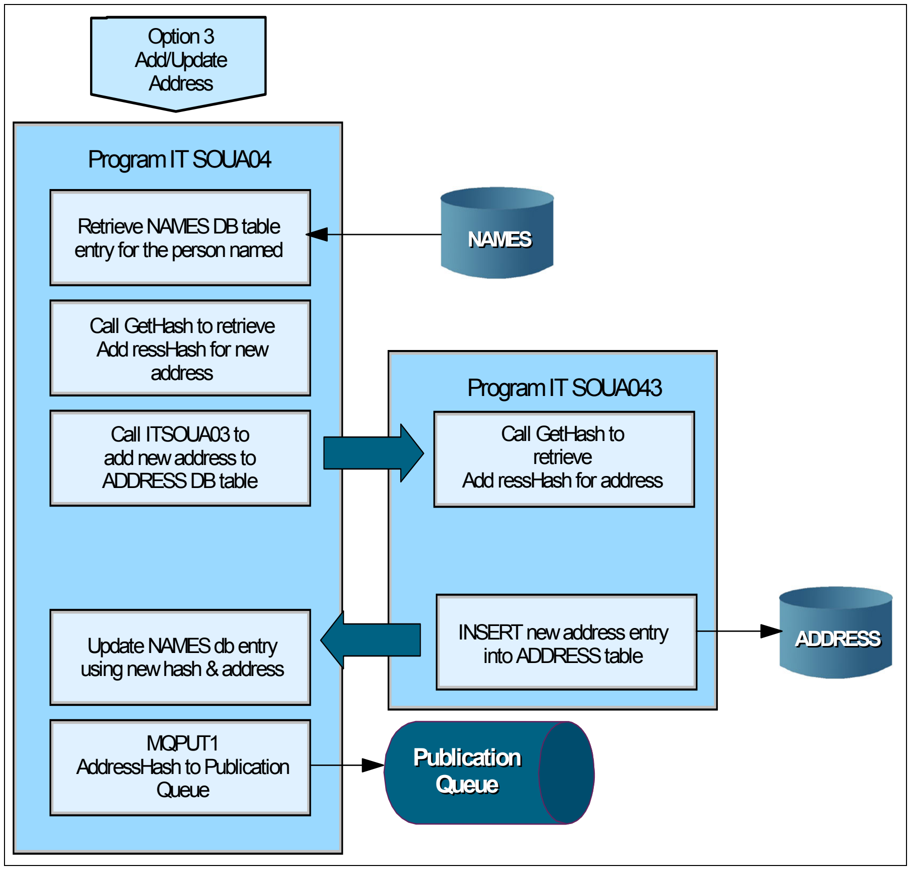
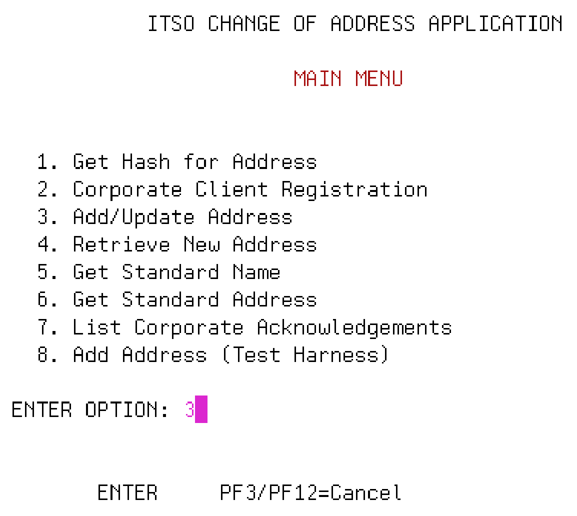

Option 3
Add/Update
Address
Program IT SOUA04
Retrieve NAMES DB table entry for the person named
Call GetHash to retrieve Add ressHash for new address
Call ITSOUA03 to add new address to ADDRESS DB table
Update NAMES db entry using new hash & address
MQPUT1 AddressHash to Publication Queue
NAMES
Program IT SOUA043
Call GetHash to retrieve Add ressHash for address
INSERT new address entry into ADDRESS table
ADDRESS
Publication Queue

Figure 9-3 Flow of the Address Update CICS programs ITSOUA04/03

9.2.1 The trace through CICS

This section shows a trace through one scenario of the Change of Address application.

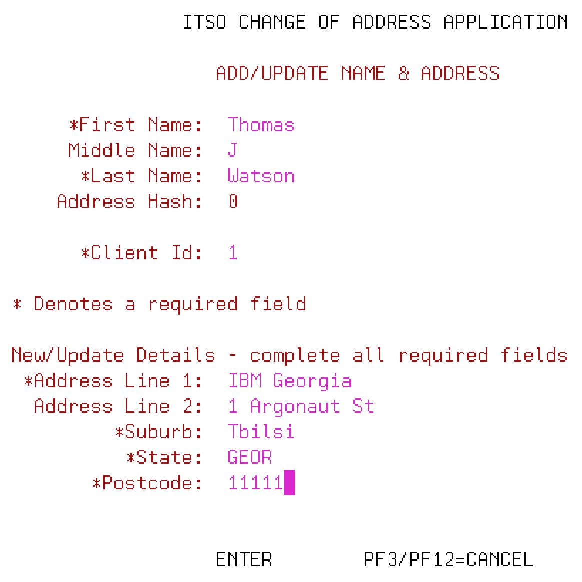
The trace begins with the postal service operator invoking the Address Change option by selecting Option 3 from the IT00 main menu. The sequence of events from a CICS perspective involves three maps and is seen in Figure 9-4 on page 221, Figure 9-5 on page 221, and Figure 9-6 on page 222.



```
               ITSO CHANGE OF ADDRESS APPLICATION

                          MAIN MENU


      1. Get Hash for Address
      2. Corporate Client Registration
      3. Add/Update Address
      4. Retrieve New Address
      5. Get Standard Name
      6. Get Standard Address
      7. List Corporate Acknowledgements
      8. Add Address (Test Harness)

    ENTER OPTION: 3


           ENTER     PF3/PF12=Cancel
```

Figure 9-4   Option 3 to Add/Update Address

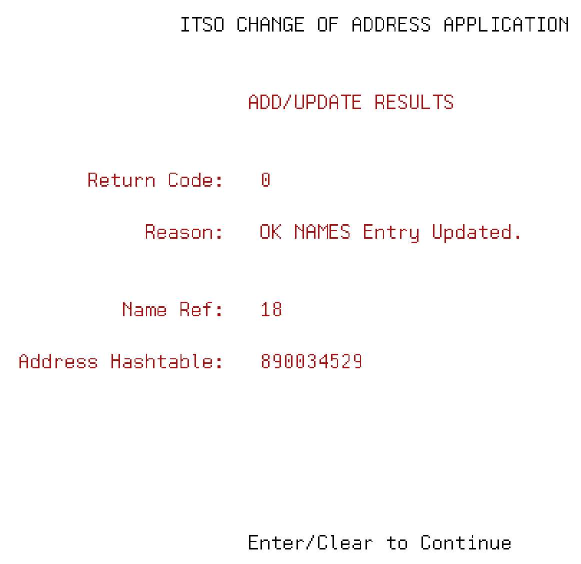
The operator then enters the new address details:



```
                 ITSO CHANGE OF ADDRESS APPLICATION

                      ADD/UPDATE NAME & ADDRESS

        *First Name:  Thomas
       Middle Name:  J
         *Last Name:  Watson
      Address Hash:  0

         *Client Id:  1

  * Denotes a required field

  New/Update Details - complete all required fields
   *Address Line 1:  IBM Georgia
    Address Line 2:  1 Argonaut St
           *Suburb:  Tbilsi
            *State:  GEOR
         *Postcode:  11111

                      ENTER       PF3/PF12=CANCEL
```

Figure 9-5   Entering new address details

Next the operator presses **ENTER** to perform the update.



```
                ITSO CHANGE OF ADDRESS APPLICATION

                      ADD/UPDATE RESULTS

        Return Code:   0

             Reason:   OK NAMES Entry Updated.

           Name Ref:   18

  Address Hashtable:   890034529



                     Enter/Clear to Continue
```

Figure 9-6 New address successfully entered

The flowchart in Figure 9-3 on page 220 describes the action of this update process and names the CICS business logic programs that are invoked for this scenario:

- **ITSOUA04** - updates the new address in a local database and publishes the notification of the update to the broker using MQPUT.
- **ITSOUA03** - INSERTs the new address into a database.

1. Start the transaction

The CICS trace in Example 9-3 on page 223 shows the start of program ITSOUA01, which SENDs the entered name and new address details from Figure 9-5 on page 221 and then returns with transaction IT14.

Example 9-3 Starting the transaction

```
AP 19A0 APLX  ENTRY START_PROGRAM        ITSOUA01,CEDF,FULLAPI,EXEC,NO,2402BB58,00000000 , 00000000,1,NO
DS 0010 DSBR  ENTRY INQUIRE_TCB
DS 0011 DSBR  EXIT  INQUIRE_TCB/OK       22C16D40
AP E160 EXEC  ENTRY SEND                 'MAP1    ' AT X'26704808',AT X'267047F0','ITSOMS2' AT X'A67047FC',MAPO
AP 00FA BMS   ENTRY SEND-OUT             SAVE ERASE MAP MAPSET       0003,00000562 ....,04000020 ....
AP 00FA BMS   ENTRY SEND-OUT             SAVE ERASE MAP MAPSET       0003,00000562 ....,04000020 ....
AP FD01 ZARQ  ENTRY APPL_REQ             24FDE030,ERASE,WRITE
AP FD81 ZARQ  EXIT  APPL_REQ
AP 00FA BMS   EXIT                                                   0005,00000000 ....,00000000 ....
AP 00FA BMS   EXIT                                                   0005,00000000 ....,00000000 ....
AP E161 EXEC  EXIT  SEND                 'MAP1    ' AT X'26704808',AT X'267047F0','ITSOMS2' AT X'A67047FC',MAPO
AP E160 EXEC  ENTRY RETURN               'IT14' AT X'A6704828',NOHANDLE,C/370,00005000
AP 1700 TFIQ  ENTRY SET_TERMINAL_FACILITY IT14,NO
AP 1701 TFIQ  EXIT  SET_TERMINAL_FACILITY/OK
DS 0010 DSBR  ENTRY INQUIRE_TASK
DS 0011 DSBR  EXIT  INQUIRE_TASK/OK      24EAE030 , 00000003
DS 0002 DSAT  ENTRY RELEASE_OPEN_TCB     24EAE030 , 00000003
DS 0003 DSAT  EXIT  RELEASE_OPEN_TCB/OK
AP 19A1 APLX  EXIT  START_PROGRAM/OK     ,NO,ITSOUA01
AP 2500 ERMSP ENTRY PERFORM_PREPARE      NO,0005C864
AP 2501 ERMSP EXIT  PERFORM_PREPARE/OK   READ_ONLY
AP 1760 LTRC  ENTRY PERFORM_PREPARE      NO,24FDE030
AP 1761 LTRC  EXIT  PERFORM_PREPARE/OK   READ_ONLY
AP 05A8 APRC  ENTRY PERFORM_PREPARE      NO,00000001
AP 05A9 APRC  EXIT  PERFORM_PREPARE/OK   READ_ONLY
AP 2500 ERMSP ENTRY PERFORM_COMMIT       00131730,NO,NO,NO,NO,NO,FORWARD,ABORT,UNNECESSARY,EYU9XSTR
AP 2501 ERMSP EXIT  PERFORM_COMMIT/OK    YES,YES,YES,NO,UNNECESSARY,EYU9XSTR
AP 2500 ERMSP ENTRY PERFORM_COMMIT       NO,FORWARD,0005C864
AP 2520 ERM   ENTRY CALL-TRUES-FOR-TASK-END
AP 2521 ERM   EXIT  CALL-TRUES-FOR-TASK-END
AP 2501 ERMSP EXIT  PERFORM_COMMIT/OK    YES
AP 1760 LTRC  ENTRY PERFORM_COMMIT       NO,FORWARD,24FDE030
AP 1710 TFRF  ENTRY RELEASE_FACILITY     NO,NORMAL,24FDE030,TC05
AP FD0B ZISP  ENTRY FACILITY_REQ         24FDE030,FREE_DETACH,IMPLICIT
XM 1001 XMIQ  ENTRY SET_TRANSACTION      NONE,NO
XM 1002 XMIQ  EXIT  SET_TRANSACTION/OK
AP FD03 ZDET  ENTRY DETACH               24FDE030,TC05
AP FD18 ZSDS  ENTRY SEND_DFSYN           24FDE030,TC05
AP FD1D ZSDR  ENTRY SEND_DFSYN_RESP      24FDE030,TC05
AP FC90 VIO   EVENT TCTTE(24FDE030)      SC38TC05,0032,SEND,DATA,0,RQE1,OIC,EB
AP FD8B ZISP  EXIT  FACILITY_REQ
AP 1711 TFRF  EXIT  RELEASE_FACILITY/OK  TC05
AP 1761 LTRC  EXIT  PERFORM_COMMIT/OK    NO
AP 05A8 APRC  ENTRY PERFORM_COMMIT       NO,FORWARD,00000001
AP 05A9 APRC  EXIT  PERFORM_COMMIT/OK    NO
AP 0590 APXM  ENTRY RELEASE_XM_CLIENT    NORMAL
AP 0591 APXM  EXIT  RELEASE_XM_CLIENT/OK
```

2. Call DB2 and generate an AddressHash

Transaction IT14 starts program ITSOUA02, which receives the entered data into a commarea and then LINKS to the business logic module ITSOUA04. This program performs the following and is highlighted in Example 9-4 on page 224:

– Call DB2 to retrieve the record for the person from the local database.

– Link to program ITSOGH03 to generate a hash value for the new address.

– Link to program ITSOUA03.

Example 9-4 Check that name exists, and generate a Hash value

```
AP 0590 APXM  ENTRY INIT_XM_CLIENT        YES
AP EA00 TMP   ENTRY LOCATE                PFT,DFHCICST
AP EA01 TMP   EXIT  LOCATE                PFT,DFHCICST,22C16B80,NORMAL
AP 0591 APXM  EXIT  INIT_XM_CLIENT/OK
AP 1790 TFXM  ENTRY INIT_XM_CLIENT        24FDE030 , 02500000
XM 1001 XMIQ  ENTRY SET_TRANSACTION       TERMINAL,24FDE030
XM 1002 XMIQ  EXIT  SET_TRANSACTION/OK
AP 1791 TFXM  EXIT  INIT_XM_CLIENT/OK     00000000,00000000,YES,NO
DS 0002 DSAT  ENTRY SET_PRIORITY          1
DS 0003 DSAT  EXIT  SET_PRIORITY/OK
AP 0590 APXM  ENTRY BIND_XM_CLIENT
AP 0591 APXM  EXIT  BIND_XM_CLIENT/OK
AP 1790 TFXM  ENTRY BIND_XM_CLIENT        24FDE030 , 02500000
AP 1791 TFXM  EXIT  BIND_XM_CLIENT/OK     YES,ITSOUA02,YES
AP 0590 APXM  ENTRY RMI_START_OF_TASK
AP 2520 ERM   ENTRY CALL-TRUES-FOR-TASK-START
AP 2521 ERM   EXIT  CALL-TRUES-FOR-TASK-START
AP 0591 APXM  EXIT  RMI_START_OF_TASK/OK
AP 19A0 APLX  ENTRY START_PROGRAM         ITSOUA02,CEDF,FULLAPI,EXEC,NO,2402BBE0,00000000 , 00000000,1,NO
DS 0010 DSBR  ENTRY INQUIRE_TCB
DS 0011 DSBR  EXIT  INQUIRE_TCB/OK        22C16D40
AP E160 EXEC  ENTRY RECEIVE               'MAP1    ' AT X'26706774',AT X'26708130','ITSOMS2' AT X'A670676C',TERM
AP 00FA BMS   ENTRY MAP-FROM              IN MAP MAPSET               0003,00020505 ....,00000020 ....
AP 00FA BMS   ENTRY MAP-FROM              IN MAP MAPSET               0003,00020505 ....,00000020 ....
AP 00FA BMS   EXIT                                                    0005,00000000 ....,00000000 ....
AP 00FA BMS   EXIT                                                    0005,00000000 ....,00000000 ....
AP E161 EXEC  EXIT  RECEIVE               'MAP1    ' AT X'26706774',AT X'26708130','ITSOMS2' AT X'A670676C',TERM
AP E160 EXEC  ENTRY ADDRESS               AT X'A67084D8',NOHANDLE,C/370,00350003
AP E161 EXEC  EXIT  ADDRESS               X'002000D0' AT X'A67084D8',0,0,NOHANDLE,C/370,00350003
AP E160 EXEC  ENTRY LINK                  'ITSOUA04' AT X'267067D0','Thomas                                   ..
AP 19A0 APLX  ENTRY START_PROGRAM         ITSOUA04,CEDF,FULLAPI,EXEC,NO,2402BC24,26707DF8 , 000001DC,2,NO
DS 0010 DSBR  ENTRY INQUIRE_TCB
DS 0011 DSBR  EXIT  INQUIRE_TCB/OK        22C16DD0
AP E160 EXEC  ENTRY ADDRESS               AT X'A67930D8',NOHANDLE,C/370,00019500
AP E161 EXEC  EXIT  ADDRESS               X'26707DF8' AT X'A67930D8',0,0,NOHANDLE,C/370,00019500
AP E160 EXEC  ENTRY WRITEQ                TS 'UA04    ' AT X'26790B10','Thom' AT X'267D7CD0',4 AT X'A6793388',A
AP E161 EXEC  EXIT  WRITEQ                TS 'UA04    ' AT X'26790B10','Thom' AT X'267D7CD0',4 AT X'A6793388',A
AP 2520 ERM   ENTRY C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
AP 3180 D2EX1 ENTRY APPLICATION           REQUEST EXEC SQL SELECT
AP 3250 D2D2  ENTRY DB2_API_CALL          24039030
AP 3251 D2D2  EXIT  DB2_API_CALL/OK
AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXEC SQL SELECT
AP 2521 ERM   EXIT  C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
AP E160 EXEC  ENTRY LINK                  'ITSOGH03' AT X'26790BC8','IBM Georgia                            ..
AP 19A0 APLX  ENTRY START_PROGRAM         ITSOGH03,CEDF,FULLAPI,EXEC,NO,2402BC68,26792F90 , 00000144,3,NO
DS 0010 DSBR  ENTRY INQUIRE_TCB
DS 0011 DSBR  EXIT  INQUIRE_TCB/OK        22C16E60
AP E160 EXEC  ENTRY ADDRESS               AT X'A67F8898',NOHANDLE,C/370,00014012
AP E161 EXEC  EXIT  ADDRESS               X'26792F90' AT X'A67F8898',0,0,NOHANDLE,C/370,00014012
AP E160 EXEC  ENTRY WRITEQ                TS 'FRED    ' AT X'267F72A4','IBM Georgia
AP E161 EXEC  EXIT  WRITEQ                TS 'FRED    ' AT X'267F72A4','IBM Georgia
AP E160 EXEC  ENTRY RETURN                NOHANDLE C/370 00017610
DS 0010 DSBR  ENTRY INQUIRE_TASK
DS 0011 DSBR  EXIT  INQUIRE_TASK/OK       22C03D50 , 00000007
DS 0004 DSSR  ENTRY RESUME                005B0003
DS 0005 DSSR  EXIT  RESUME/OK
DS 0002 DSAT  ENTRY RELEASE_OPEN_TCB      22C03D50 , 00000007
DS 0003 DSAT  EXIT  RELEASE_OPEN_TCB/OK
AP 19A1 APLX  EXIT  START_PROGRAM/OK      ,NO,ITSOGH03
AP E161 EXEC  EXIT  LINK                  'ITSOGH03' AT X'26790BC8','IBM Georgia                            ..
AP E160 EXEC  ENTRY LINK                  'ITSOUA03' AT X'26790B44','IBM Georgia                            ..
```

3. Update database with new address

Program ITSOUA03 now adds the new address and new hash value to the local database. Example 9-5 shows that the following is performed:

- ITSOUA03 starts.
- Link to ITSOGH03 to GetHash value.
- Call to DB2 to INSERT new address and hash value into the database.
- RETURN to ITSOUA04.

Example 9-5 Update new address

```
AP 19A0 APLX  ENTRY START_PROGRAM        ITSOUA03,CEDF,FULLAPI,EXEC,NO,2402BCAC,267930E0 , 00000140,3,NO
DS 0005 DSSR  EXIT  SUSPEND/OK
AP F300 APTI  ENTRY NOTIFY               00E70000 , 00000000
DS 0004 DSSR  ENTRY RESUME               00810003
DS 0005 DSSR  EXIT  RESUME/OK
AP F301 APTI  EXIT  NOTIFY/OK
DS 0004 DSSR  ENTRY SUSPEND              005B0003,TIEXPIRY,NO,TIMER,DS_NUDGE
DS 0005 DSSR  EXIT  SUSPEND/OK
AP F322 APTIX RESUM SYSTEM               TASK APTIX RESUMED
AP 00F3 ICP   ENTRY ICE                  EXPIRY ANALYSIS             C003,00000000 ....,00000000 ....
AP 00F3 ICP   EXIT  NORMAL                                           0005,00000000 ....,00000000 ....
DS 0004 DSSR  ENTRY SUSPEND              00810003,ICEXPIRY,NO,TIMER,DFHAPTIX
DS 0005 DSSR  EXIT  WAIT_MVS/OK
AP E161 EXEC  EXIT  WAIT                 EXTERNAL X'250E6ABC' AT X'250E65F8',3 AT X'250E662C','CDB2TIME' AT X'
AP E160 EXEC  ENTRY ENQ                  '.{.0....' AT X'240552B0',8 AT X'A50E661C',NOHANDLE,ASM
AP E161 EXEC  EXIT  ENQ                  '.{.0....' AT X'240552B0',8 AT X'A50E661C',0,0,NOHANDLE,ASM
DS 0010 DSBR  ENTRY INQUIRE_TCB
DS 0011 DSBR  EXIT  INQUIRE_TCB/OK       22C16E60
AP E160 EXEC  ENTRY ADDRESS              AT X'A67FA830',NOHANDLE,C/370,00008001
AP E161 EXEC  EXIT  ADDRESS              X'267930E0' AT X'A67FA830',0,0,NOHANDLE,C/370,00008001
AP E160 EXEC  ENTRY WRITEQ               TS 'UA03    ' AT X'267F9024','IBM Georgia
AP E161 EXEC  EXIT  WRITEQ               TS 'UA03    ' AT X'267F9024','IBM Georgia
AP E160 EXEC  ENTRY LINK                 'ITSOGH03' AT X'267F9044','IBM Georgia                          ..
AP 19A0 APLX  ENTRY START_PROGRAM        ITSOGH03,CEDF,FULLAPI,EXEC,NO,2402BC68,267FA838 , 00000144,4,NO
DS 0010 DSBR  ENTRY INQUIRE_TCB
DS 0011 DSBR  EXIT  INQUIRE_TCB/OK       22C16EF0
AP E160 EXEC  ENTRY ADDRESS              AT X'A67F8898',NOHANDLE,C/370,00014012
AP E161 EXEC  EXIT  ADDRESS              X'267FA838' AT X'A67F8898',0,0,NOHANDLE,C/370,00014012
AP E160 EXEC  ENTRY WRITEQ               TS 'FRED    ' AT X'267F72A4','IBM Georgia
AP E161 EXEC  EXIT  WRITEQ               TS 'FRED    ' AT X'267F72A4','IBM Georgia
AP E160 EXEC  ENTRY RETURN               NOHANDLE C/370 00017610
DS 0010 DSBR  ENTRY INQUIRE_TASK
DS 0011 DSBR  EXIT  INQUIRE_TASK/OK      22C03C38 , 00000007
DS 0002 DSAT  ENTRY RELEASE_OPEN_TCB     22C03C38 , 00000007
DS 0003 DSAT  EXIT  RELEASE_OPEN_TCB/OK
AP 19A1 APLX  EXIT  START_PROGRAM/OK     ,NO,ITSOGH03
AP E161 EXEC  EXIT  LINK                 'ITSOGH03' AT X'267F9044','IBM Georgia                          ..
AP 2520 ERM   ENTRY C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL INSERT
AP 3250 D2D2  ENTRY DB2_API_CALL         24039030
AP 3251 D2D2  EXIT  DB2_API_CALL/OK
AP 3181 D2EX1 EXIT  APPLICATION-REQUEST  SQLCODE 0 RETURNED ON EXEC SQL INSERT
AP 2521 ERM   EXIT  C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
AP E160 EXEC  ENTRY RETURN               NOHANDLE C/370 00018800
AP 19A1 APLX  EXIT  START_PROGRAM/OK     ,NO,ITSOUA03
AP E161 EXEC  EXIT  LINK                 'ITSOUA03' AT X'26790B44','IBM Georgia                          ..
```

4. Call MQ to write the old AddressHash to a queue

The processing has now returned to ITSOUA04, which performs the following actions and is highlighted in Example 9-6:

- Call to DB2 to UPDATE the new address and hash in the NAMES table.
- Call to MQ to MQPUT the Old AddressHash value to the queue named CICSWSAP.PUBLICATION.QUEUE.
- Return to presentation logic program ITSOUA02.

Example 9-6 Call to MQ to write the Old AddressHash

```
AP 2520 ERM   ENTRY C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
AP 3180 D2EX1 ENTRY APPLICATION           REQUEST EXEC SQL UPDATE
AP 3250 D2D2  ENTRY DB2_API_CALL          24039030
AP 3251 D2D2  EXIT  DB2_API_CALL/OK
AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXEC SQL UPDATE
AP 2521 ERM   EXIT  C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
AP 2520 ERM   ENTRY C-APPLICATION-CALL-TO-TRUE(MQM     )
AP E160 EXEC  ENTRY ASSIGN                AT X'A6E01977',NOHANDLE,ASM,15920000
AP E161 EXEC  EXIT  ASSIGN                X'40' AT X'A6E01977',0,0,NOHANDLE,ASM,15920000
AP E160 EXEC  ENTRY ASSIGN                AT X'A50F4530',NOHANDLE,ASM,16600000
AP E161 EXEC  EXIT  ASSIGN                'CICSUSER' AT X'A50F4530',0,0,NOHANDLE,ASM,16600000
AP E160 EXEC  ENTRY ENTER                 0 AT X'24ED7550','CICSWSAP.PUBLICATION.QUEUE......................' A
AP 0000 USER  EVENT APPLICATION-PROGRAM-ENTRY CSQCP1ON CICSWSAP.PUBLICATION.QUEUE.....................
AP E161 EXEC  EXIT  ENTER                 0 AT X'24ED7550','CICSWSAP.PUBLICATION.QUEUE......................' A
AP E160 EXEC  ENTRY ENTER                 0 AT X'24ED7550',' 01387662669' AT X'267D7848',12 AT X'26E01970','CSQ
AP 0000 USER  EVENT APPLICATION-PROGRAM-ENTRY CSQCP1MD 01387662669
AP E161 EXEC  EXIT  ENTER                 0 AT X'24ED7550',' 01387662669' AT X'267D7848',12 AT X'26E01970','CSQ
DS 0004 DSSR  ENTRY WAIT_MVS              MQSeries,250F4580,NO,TASKSWCH
DS 0005 DSSR  EXIT  WAIT_MVS/OK
AP E160 EXEC  ENTRY ENTER                 0 AT X'24ED7550','CSQ MQ8G        {.....s.........................' A
AP 0000 USER  EVENT APPLICATION-PROGRAM-ENTRY CSQCP1MI CSQ MQ8G {.....s.........................
AP E161 EXEC  EXIT  ENTER                 0 AT X'24ED7550','CSQ MQ8G        {.....s.........................' A
AP 2521 ERM   EXIT  C-APPLICATION-CALL-TO-TRUE(MQM     )
AP E160 EXEC  ENTRY RETURN                NOHANDLE C/370 00079900
DS 0010 DSBR  ENTRY INQUIRE_TASK
DS 0011 DSBR  EXIT  INQUIRE_TASK/OK       24EAE378 , 00000007
DS 0002 DSAT  ENTRY RELEASE_OPEN_TCB      24EAE378 , 00000007
DS 0003 DSAT  EXIT  RELEASE_OPEN_TCB/OK
AP 19A1 APLX  EXIT  START_PROGRAM/OK      ,NO,ITSOUA04
AP E161 EXEC  EXIT  LINK                  'ITSOUA04' AT X'267067D0','Thomas                                 ..
```

5. Display results in CICS and terminate task

This is the conclusion of the CICS processing with a return to the presentation logic in program ITSOUA02 and task cleanup. Example 9-7 on page 227 shows the following actions:

- ITSOUA02 SENDs the map containing the results of the Add/Update Address to the terminal. This is also seen in Figure 9-6 on page 222.
- The CICS task is then terminated.

Example 9-7 Results are displayed to the terminal, and the CICS task is terminated

```
AP E160 EXEC  ENTRY SEND                 'MAP2   ' AT X'267067F0',AT X'26707FD8','ITSOMS2' AT X'A670676C',TERM
AP 00FA BMS   ENTRY SEND-OUT             SAVE ERASE MAP MAPSET       0003,000005E2 ...S,04000020 ....
AP 00FA BMS   ENTRY SEND-OUT             SAVE ERASE MAP MAPSET       0003,000005E2 ...S,04000020 ....
AP FD01 ZARQ  ENTRY APPL_REQ             24FDE030,ERASE,WRITE
AP FD81 ZARQ  EXIT  APPL_REQ
AP 00FA BMS   EXIT                                                   0005,00000000 ....,00000000 ....
AP 00FA BMS   EXIT                                                   0005,00000000 ....,00000000 ....
AP E161 EXEC  EXIT  SEND                 'MAP2   ' AT X'267067F0',AT X'26707FD8','ITSOMS2' AT X'A670676C',TERM
AP E160 EXEC  ENTRY RETURN               'IT00' AT X'A67067B4',NOHANDLE,C/370,00770003
AP 1700 TFIQ  ENTRY SET_TERMINAL_FACILITY IT00,NO
AP 1701 TFIQ  EXIT  SET_TERMINAL_FACILITY/OK
DS 0010 DSBR  ENTRY INQUIRE_TASK
DS 0011 DSBR  EXIT  INQUIRE_TASK/OK       24EAE030 , 00000003
DS 0002 DSAT  ENTRY RELEASE_OPEN_TCB      24EAE030 , 00000003
DS 0003 DSAT  EXIT  RELEASE_OPEN_TCB/OK
AP 19A1 APLX  EXIT  START_PROGRAM/OK      ,NO,ITSOUA02

AP FD03 ZDET  ENTRY DETACH               24FDE030,TC05
AP FD18 ZSDS  ENTRY SEND_DFSYN           24FDE030,TC05
AP FD1D ZSDR  ENTRY SEND_DFSYN_RESP      24FDE030,TC05
AP FC90 VIO   EVENT TCTTE(24FDE030)      SC38TC05,0033,SEND,DATA,0,RQE1,OIC,EB
AP FD8B ZISP  EXIT  FACILITY_REQ
AP 1711 TFRF  EXIT  RELEASE_FACILITY/OK   TC05
AP 1761 LTRC  EXIT  PERFORM_COMMIT/OK     NO
AP 05A8 APRC  ENTRY PERFORM_COMMIT        NO,FORWARD,00000001
AP 05A9 APRC  EXIT  PERFORM_COMMIT/OK     NO
AP 0590 APXM  ENTRY RELEASE_XM_CLIENT     NORMAL
AP 0591 APXM  EXIT  RELEASE_XM_CLIENT/OK
```

9.2.2 Tracing the Broker on System z

The flowchart in Figure 9-7 on page 228 shows the Hash Publication message flow.

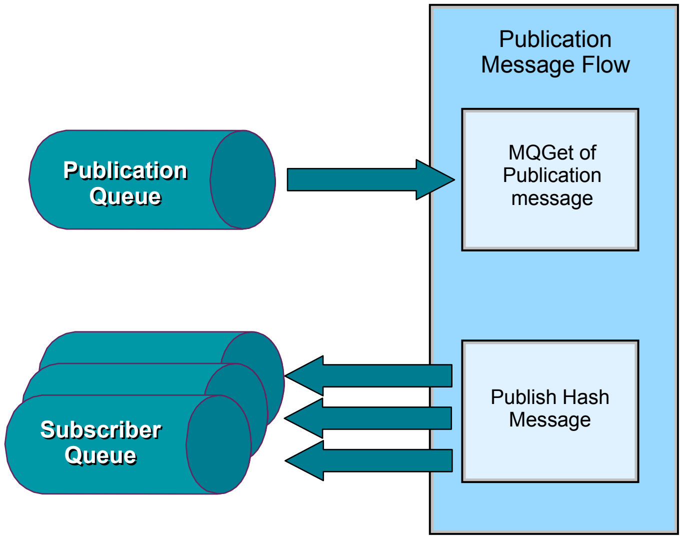


Figure 9-7 The Hash Publication Flow

Example 9-8 The HashPublication message flow trace

```
Timestamps are formatted in local time, 300 minutes before GMT.
Trace written by version ; formatter version 6002
------------------------------------------------------------------------------------------
The following is the MQInput node processing the Hash message from the publication queue CICSWSAP.PUBLICATION.QUEUE
------------------------------------------------------------------------------------------
2007-02-21 22:39:58.438136      23   UserTrace   BIP2632I: Message received and propagated to 'out' terminal of MQ input
node 'HashPublication.Get Pub Msg'.
2007-02-21 22:39:58.438228      23   UserTrace   BIP6060I: Parser type ''Properties'' created on behalf of node
'HashPublication.Get Pub Msg' to handle portion of incoming message of length 0 bytes beginning at offset '0'.
2007-02-21 22:39:58.438308      23   UserTrace   BIP6061I: Parser type ''MQMD'' created on behalf of node
'HashPublication.Get Pub Msg' to handle portion of incoming message of length '364' bytes beginning at offset '0'. Parser
type selected based on value ''MQHMD'' from previous parser.
2007-02-21 22:39:58.438400      23   UserTrace   BIP6061I: Parser type ''MRM'' created on behalf of node
'HashPublication.Get Pub Msg' to handle portion of incoming message of length '12' bytes beginning at offset '364'. Parser
type selected based on value ''MRM'' from previous parser.
------------------------------------------------------------------------------------------
Having successfully parsed the Hash message, the Publish node is invoked to distribute the message to all registered
subscribers. At this time there were 3 subs:
------------------------------------------------------------------------------------------
2007-02-21 22:39:58.438460      23   UserTrace   BIP7080I: Node 'HashPublication.Publish Hash.ComIbmPSService': The
Publication Node with Subscription Point ''AddressChange'' has received a message of type 'Publish'.
The Publication Node with Subscription Point ''AddressChange'' has started processing a message.
No user action required.
2007-02-21 22:39:58.438816      23   UserTrace   BIP7081I: The Publication Node has matched '3' subscriptions to topic
''CICSWSAP/AddressChange'' for subscription point ''AddressChange''.
The Publication Node has matched subscriptions for the current publication and topic.
No user action required.
2007-02-21 22:39:58.438868      23   UserTrace   BIP7082I: Node 'HashPublication.Publish Hash.ComIbmPSService':
Publishing to destination 'MQ8G:CICSWSAP.ADDRESS.CHANGE.VBS' for user 'ANDREWG'.
A publication destination is being added to the list of destinations to 'MQ8G:CICSWSAP.ADDRESS.CHANGE.VBS' for user
'ANDREWG'.
No user action required.
2007-02-21 22:39:58.438900      23   UserTrace   BIP7082I: Node 'HashPublication.Publish Hash.ComIbmPSService':
Publishing to destination 'MQ8G:CICSWSAP.ADDRESS.CHANGE.MF' for user 'ANDREWG'.
```

```
A publication destination is being added to the list of destinations to 'MQ8G:CICSWSAP.ADDRESS.CHANGE.MF' for user
'ANDREWG'.
No user action required.
2007-02-21 22:39:58.438912       23   UserTrace   BIP7082I: Node 'HashPublication.Publish Hash.ComIbmPSService':
Publishing to destination 'MQ8G:CICSWSAP.ADDRESS.CHANGE.WIN' for user 'ANDREWG'.
A publication destination is being added to the list of destinations to 'MQ8G:CICSWSAP.ADDRESS.CHANGE.WIN' for user
'ANDREWG'.
No user action required.
2007-02-21 22:39:58.438936       23   UserTrace   BIP7085I: Node 'HashPublication.Publish Hash.ComIbmPSService': The
Publication Node has propagated a message to its output terminal for subscription point ''AddressChange''.
The Publication Node has propagated the current message to its output terminal.
No user action required.
-----------------------------------------------------------------------------------------------
```
The Publish node now does an MQPUT1 to each subscription queue:
```
-----------------------------------------------------------------------------------------------
2007-02-21 22:39:58.439188       23   UserTrace   BIP2638I: The MQ output node 'HashPublication.Publish
Hash.ComIbmMQOutput' attempted to write a message to queue ''CICSWSAP.ADDRESS.CHANGE.VBS'' connected to queue manager
''MQ8G''. The MQCC was '0' and the MQRC was '0'.
2007-02-21 22:39:58.439200       23   UserTrace   BIP2622I: Message successfully output by output node
'HashPublication.Publish Hash.ComIbmMQOutput' to queue ''CICSWSAP.ADDRESS.CHANGE.VBS'' on queue manager ''MQ8G''.
2007-02-21 22:39:58.439336       23   UserTrace   BIP2638I: The MQ output node 'HashPublication.Publish
Hash.ComIbmMQOutput' attempted to write a message to queue ''CICSWSAP.ADDRESS.CHANGE.MF'' connected to queue manager
''MQ8G''. The MQCC was '0' and the MQRC was '0'.
2007-02-21 22:39:58.439348       23   UserTrace   BIP2622I: Message successfully output by output node
'HashPublication.Publish Hash.ComIbmMQOutput' to queue ''CICSWSAP.ADDRESS.CHANGE.MF'' on queue manager ''MQ8G''.
2007-02-21 22:39:58.439512       23   UserTrace   BIP2638I: The MQ output node 'HashPublication.Publish
Hash.ComIbmMQOutput' attempted to write a message to queue ''CICSWSAP.ADDRESS.CHANGE.WIN'' connected to queue manager
''MQ8G''. The MQCC was '0' and the MQRC was '0'.
2007-02-21 22:39:58.439556       23   UserTrace   BIP2622I: Message successfully output by output node
'HashPublication.Publish Hash.ComIbmMQOutput' to queue ''CICSWSAP.ADDRESS.CHANGE.WIN'' on queue manager ''MQ8G''.
-----------------------------------------------------------------------------------------------
```
All done.

9.2.3 Tracing the Broker on Windows

Now the RetrieveAddressWithDB message flow is triggered on the Windows platform by the receipt of the hash publication.

Example 9-9 RetrieveAddressWithDB MsgFlow Pt1

The RetrieveAddressWithDB msgflow is triggered by the arrival of the hashcode on our subscription queue CICSWSAP.ADDRESS.CHANGE.
The MQInput node retrives the message and parses it.
```
-----------------------------------------------------------------------------------------------

2007-02-22 14:40:00.264122    10140   UserTrace   BIP2632I: Message received and propagated to 'out' terminal of MQ input
node 'RetrieveAddressWithDB.GetPublication'.
2007-02-22 14:40:00.264232    10140   UserTrace   BIP6060I: Parser type ''Properties'' created on behalf of node
'RetrieveAddressWithDB.GetPublication' to handle portion of incoming message of length 0 bytes beginning at offset '0'.
2007-02-22 14:40:00.264292    10140   UserTrace   BIP6061I: Parser type ''MQMD'' created on behalf of node
'RetrieveAddressWithDB.GetPublication' to handle portion of incoming message of length '364' bytes beginning at offset
'0'. Parser type selected based on value ''MQHMD'' from previous parser.
2007-02-22 14:40:00.264479    10140   UserTrace   BIP6061I: Parser type ''MRM'' created on behalf of node
'RetrieveAddressWithDB.GetPublication' to handle portion of incoming message of length '12' bytes beginning at offset
'364'. Parser type selected based on value ''MRM'' from previous parser.

-----------------------------------------------------------------------------------------------
```
The next node performs an SQL query to retrieve entries from the local CUSTOMER database table that match the hash code just received. Of course there may be more than one entry, but for simplicity, we only process the first entry we find.
```
-----------------------------------------------------------------------------------------------
```

```
2007-02-22 14:40:00.264560    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Query Hash Code': Executing
statement   ''DECLARE tns NAMESPACE 'http://schemas.xmlsoap.org/soap/envelope/';'' at ('.tns', '1.1').
2007-02-22 14:40:00.264744    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Query Hash Code': Executing
statement   ''DECLARE resns NAMESPACE 'http://www.ITSORA03.ITSORACA.Response.com';'' at ('.resns', '1.1').
2007-02-22 14:40:00.264800    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Query Hash Code': Executing
statement   ''DECLARE reqns NAMESPACE 'http://www.ITSORA03.ITSORACA.Request.com';'' at ('.reqns', '1.1').
2007-02-22 14:40:00.264860    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Query Hash Code': Executing
statement   ''BEGIN ... END;'' at ('.RetrieveAddressWithDB_Database.Main', '2.2').
2007-02-22 14:40:00.264907    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Query Hash Code': Executing
statement   ''SET Environment.Variables = THE (SELECT ROW (COLUMN(0) AS *:CUSTID, COLUMN(1) AS *:FIRSTNAME, COLUMN(2) AS
*:MIDDLENAME, COLUMN(3) AS *:LASTNAME, COLUMN(4) AS *:ADDRESS1, COLUMN(5) AS *:ADDRESS2, COLUMN(6) AS *:SUBURB, COLUMN(7)
AS *:STATE, COLUMN(8) AS *:POSTCODE, COLUMN(9) AS *:HASH) FROM DATABASE(, Root.MRM.hash));'' at
('.RetrieveAddressWithDB_Database.Main', '3.3').
2007-02-22 14:40:00.264948    10140   UserTrace   BIP2538I: Node 'RetrieveAddressWithDB.Query Hash Code': Evaluating
expression ''THE (SELECT ROW (COLUMN(0) AS *:CUSTID, COLUMN(1) AS *:FIRSTNAME, COLUMN(2) AS *:MIDDLENAME, COLUMN(3) AS
*:LASTNAME, COLUMN(4) AS *:ADDRESS1, COLUMN(5) AS *:ADDRESS2, COLUMN(6) AS *:SUBURB, COLUMN(7) AS *:STATE, COLUMN(8) AS
*:POSTCODE, COLUMN(9) AS *:HASH) FROM DATABASE(, Root.MRM.hash))'' at ('.RetrieveAddressWithDB_Database.Main', '4.4').
2007-02-22 14:40:00.264979    10140   UserTrace   BIP2572W: Node: 'RetrieveAddressWithDB.Query Hash Code':
('.RetrieveAddressWithDB_Database.Main', '4.4') : Finding one and only SELECT result.
2007-02-22 14:40:00.567390    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Query Hash Code': Evaluating
expression ''Root.MRM.hash'' at ('.RetrieveAddressWithDB_Database.Main', '6.24'). This resolved to ''Root.MRM.hash''. The
result was ''' 01387662669'''.
2007-02-22 14:40:00.567452    10140   UserTrace   BIP2544I: Node 'RetrieveAddressWithDB.Query Hash Code': Executing
database SQL statement ''SELECT C.CUSTID, C.FIRSTNAME, C.MIDDLENAME, C.LASTNAME, C.ADDRESS1, C.ADDRESS2, C.SUBURB,
C.STATE, C.POSTCODE, C.HASH FROM ANDREWG.CUSTOMER C WHERE (C.HASH)=(?)'' derived from ('', '1.1'); expressions
''Root.MRM.hash''; resulting parameter values ''' 01387662669'''.
2007-02-22 14:40:00.568755    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Query Hash Code': Evaluating
expression ''DATABASE(, Root.MRM.hash)'' at ('', '1.1'). This resolved to ''SELECT C.CUSTID, C.FIRSTNAME, C.MIDDLENAME,
C.LASTNAME, C.ADDRESS1, C.ADDRESS2, C.SUBURB, C.STATE, C.POSTCODE, C.HASH FROM ANDREWG.CUSTOMER C WHERE (C.HASH)=(?)''.
The result was ''Complex result''.
2007-02-22 14:40:00.569040   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
''1'' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569087   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''Thomas''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569126   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''J''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569164   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''Watson''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569212   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''IBM Gabon           ''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569262   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''1 Bantu St          ''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569302   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''Libreville          ''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569339   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''GAB       ''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569378   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
'''11111     ''' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569414   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Query Hash Code': Assigning value
''1387662669'' to field / variable ''Environment.Variables''.
2007-02-22 14:40:00.569500    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Query Hash Code': Executing
statement   ''RETURN TRUE;'' at ('.RetrieveAddressWithDB_Database.Main', '7.3').
2007-02-22 14:40:00.569558    10140   UserTrace   BIP4007I: Message propagated to 'out' terminal of node
'RetrieveAddressWithDB.Query Hash Code'.

----------------------------------------------------------------------------------------------
We've found an entry for the client 'Thomas J Watson'. The message flow needs to create a Web service call to the
RetrieveAddress CICS w-s using the full name of our client and the just published hash value.
----------------------------------------------------------------------------------------------

2007-02-22 14:40:00.569698    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''DECLARE tns NAMESPACE 'http://schemas.xmlsoap.org/soap/envelope/';'' at ('.tns', '1.1').
2007-02-22 14:40:00.569750    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''DECLARE resns NAMESPACE 'http://www.ITSORA03.ITSORACA.Response.com';'' at ('.resns', '1.1').
2007-02-22 14:40:00.569796    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''DECLARE reqns NAMESPACE 'http://www.ITSORA03.ITSORACA.Request.com';'' at ('.reqns', '1.1').
```

```
2007-02-22 14:40:00.569841    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''BEGIN ... END;'' at ('.RetrieveAddressWithDB_Compute.Main', '2.2').
2007-02-22 14:40:00.569886    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''DECLARE SOAPENV NAMESPACE 'http://schemas.xmlsoap.org/soap/envelope/';'' at
('.RetrieveAddressWithDB_Compute.Main', '3.3').
2007-02-22 14:40:00.569939    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.Properties.MessageSet = 'CICSWSAPWSDLMsgSet';'' at ('.RetrieveAddressWithDB_Compute.Main',
'5.4').
2007-02-22 14:40:00.569992    10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''CICSWSAPWSDLMsgSet''' to field / variable ''OutputRoot.Properties.MessageSet''.
2007-02-22 14:40:00.570047    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.Properties.MessageType = 'Envelope';'' at ('.RetrieveAddressWithDB_Compute.Main', '6.4').
2007-02-22 14:40:00.570093    10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''Envelope''' to field / variable ''OutputRoot.Properties.MessageType''.
2007-02-22 14:40:00.570141    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.Properties.MessageFormat = 'XML1';'' at ('.RetrieveAddressWithDB_Compute.Main', '7.4').
2007-02-22 14:40:00.570186    10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''XML1''' to field / variable ''OutputRoot.Properties.MessageFormat''.
2007-02-22 14:40:00.570245    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:FirstName =
Environment.Variables.FIRSTNAME;'' at ('.RetrieveAddressWithDB_Compute.Main', '9.3').
2007-02-22 14:40:00.570309    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Setup WS Call': Evaluating
expression ''Environment.Variables.FIRSTNAME'' at ('.RetrieveAddressWithDB_Compute.Main', '10.7'). This resolved to
''Environment.Variables.FIRSTNAME''. The result was ''ROW... Root Element Type=50331648 NameSpace='' Name='FIRSTNAME'
Value='Thomas'''.
2007-02-22 14:40:00.570508    10140   UserTrace   BIP2568I: Node 'RetrieveAddressWithDB.Setup WS Call': Copying sub-tree
from ''Environment.Variables.FIRSTNAME'' to
''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:FirstName''.
2007-02-22 14:40:00.570571    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:MiddleName =
Environment.Variables.MIDDLENAME;'' at ('.RetrieveAddressWithDB_Compute.Main', '11.3').
2007-02-22 14:40:00.570657    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Setup WS Call': Evaluating
expression ''Environment.Variables.MIDDLENAME'' at ('.RetrieveAddressWithDB_Compute.Main', '12.7'). This resolved to
''Environment.Variables.MIDDLENAME''. The result was ''ROW... Root Element Type=50331648 NameSpace='' Name='MIDDLENAME'
Value='J'''.
2007-02-22 14:40:00.570717    10140   UserTrace   BIP2568I: Node 'RetrieveAddressWithDB.Setup WS Call': Copying sub-tree
from ''Environment.Variables.MIDDLENAME'' to
''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:MiddleName''.
2007-02-22 14:40:00.570769    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:LastName =
Environment.Variables.LASTNAME;'' at ('.RetrieveAddressWithDB_Compute.Main', '13.3').
2007-02-22 14:40:00.570825    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Setup WS Call': Evaluating
expression ''Environment.Variables.LASTNAME'' at ('.RetrieveAddressWithDB_Compute.Main', '14.7'). This resolved to
''Environment.Variables.LASTNAME''. The result was ''ROW... Root Element Type=50331648 NameSpace='' Name='LASTNAME'
Value='Watson'''.
2007-02-22 14:40:00.570885    10140   UserTrace   BIP2568I: Node 'RetrieveAddressWithDB.Setup WS Call': Copying sub-tree
from ''Environment.Variables.LASTNAME'' to
''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:LastName''.
2007-02-22 14:40:00.570935    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressHash =
LTRIM(InputRoot.MRM.hash);'' at ('.RetrieveAddressWithDB_Compute.Main', '15.3').
2007-02-22 14:40:00.570986    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Setup WS Call': Evaluating
expression ''InputRoot.MRM.hash'' at ('.RetrieveAddressWithDB_Compute.Main', '16.12'). This resolved to
''InputRoot.MRM.hash''. The result was ''' 01387662669'''.
2007-02-22 14:40:00.571030    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Setup WS Call': Evaluating
expression ''LTRIM(InputRoot.MRM.hash)'' at ('.RetrieveAddressWithDB_Compute.Main', '16.6'). This resolved to ''LTRIM('
01387662669', NULL)''. The result was '''01387662669'''.
2007-02-22 14:40:00.571085    10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''01387662669''' to field / variable
''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressHash''.
2007-02-22 14:40:00.571134    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:ClientId = 1;'' at
('.RetrieveAddressWithDB_Compute.Main', '17.3').
2007-02-22 14:40:00.571201    10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
''1'' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:ClientId''.
```

```
2007-02-22 14:40:00.571259    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine1 = 'null';'' at
('.RetrieveAddressWithDB_Compute.Main', '21.3').
2007-02-22 14:40:00.571318   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''null''' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine1''.
2007-02-22 14:40:00.571367    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine2 = 'null';'' at
('.RetrieveAddressWithDB_Compute.Main', '22.3').
2007-02-22 14:40:00.571430   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''null''' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:AddressLine2''.
2007-02-22 14:40:00.571479    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Suburb = 'null';'' at
('.RetrieveAddressWithDB_Compute.Main', '23.3').
2007-02-22 14:40:00.571690   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''null''' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Suburb''.
2007-02-22 14:40:00.571743    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:State = 'null';'' at
('.RetrieveAddressWithDB_Compute.Main', '24.3').
2007-02-22 14:40:00.571803   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''null''' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:State''.
2007-02-22 14:40:00.571853    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Postcode = 'null';'' at
('.RetrieveAddressWithDB_Compute.Main', '25.3').
2007-02-22 14:40:00.571914   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''null''' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:Postcode''.
2007-02-22 14:40:00.571961    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:rc = -1;'' at
('.RetrieveAddressWithDB_Compute.Main', '26.3').
2007-02-22 14:40:00.572013    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Setup WS Call': Evaluating
expression ''-1'' at ('.RetrieveAddressWithDB_Compute.Main', '26.81'). This resolved to ''-1''. The result was ''-1''.
2007-02-22 14:40:00.572068   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
''-1'' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:rc''.
2007-02-22 14:40:00.572116    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''SET OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:reason = 'null';'' at
('.RetrieveAddressWithDB_Compute.Main', '27.3').
2007-02-22 14:40:00.572176   10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Setup WS Call': Assigning value
'''null''' to field / variable ''OutputRoot.MRM.SOAPENV:Body.reqns:ITSORA03Operation.reqns:raca.reqns:reason''.
2007-02-22 14:40:00.572225    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Setup WS Call': Executing
statement   ''RETURN TRUE;'' at ('.RetrieveAddressWithDB_Compute.Main', '29.3').
2007-02-22 14:40:00.572301    10140   UserTrace   BIP4007I: Message propagated to 'out' terminal of node
'RetrieveAddressWithDB.Setup WS Call'.

-----------------------------------------------------------------------------------------------
There is one of the following warnings for each of the elements referenced. We have edited out the others for brevity.
-----------------------------------------------------------------------------------------------

2007-02-22 14:40:00.572815    10140   UserTrace   BIP5493W: Message, element or attribute
'http://www.ITSORA03.ITSORACA.Request.com:AddressLine1' is self-defining within parent
'http://www.ITSORA03.ITSORACA.Request.com:raca'.
The message, element or attribute 'http://www.ITSORA03.ITSORACA.Request.com:AddressLine1' did not match with any
corresponding artifact in the message model hence it is considered to be self-defining. If it is not intended that this
message, element or attribute be self-defining, check that the message set is referenced in the message properties, or
modify the message model to correspond to the instance message, or modify the instance message to correspond to the
message model.


-----------------------------------------------------------------------------------------------
Now that all the necessary w-s parameters are set, we call the RetrieveAddress to invoke the actual CICS w-s.
```

9.2.4 Tracing the CICS Web service

We now pick up the trace on the CICS side where the CICS Web service pipeline processes the Web service request for the RetrieveAddress logic and hands back the result. The flowchart in Figure 9-8 shows the interaction between the message flow running in the Broker on the client Windows platform and the Web service in CICS where we are exposing our business logic as a service provider. The CICS program being called is ITSORA03.

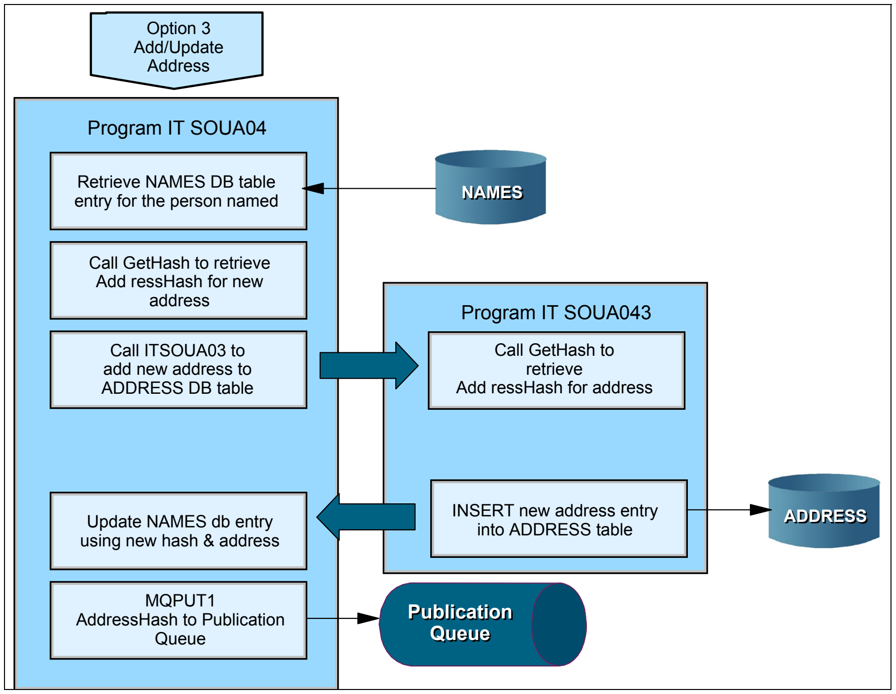

Option 3 Add/Update Address
Program IT SOUA04
Retrieve NAMES DB table entry for the person named
Call GetHash to retrieve Add ressHash for new address
Call ITSOUA03 to add new address to ADDRESS DB table
Update NAMES db entry using new hash & address
MQPUT1 AddressHash to Publication Queue
NAMES
Program IT SOUA043
Call GetHash to retrieve Add ressHash for address
INSERT new address entry into ADDRESS table
ADDRESS
Publication Queue

Figure 9-8 Client calling CICS RetrieveAddress Web service

1. The Web service request enters CICS

When the client broker submits the Web service request to the CICS region, the receiving TCPIPSERVICE detects the incoming request and the actions are performed and displayed in the trace in Example 9-10.

- Transaction CWXN starts and invokes program DFHWBXN to handle the incoming request.
- The CICS SOAP HTTP Inbound Router transaction CPIH is attached.

Example 9-10 Request for Web services enters CICS

```
SL     XM 1101 XMAT  ENTRY ATTACH                CWXN,C,NO,YES,SOCKET,24FB3000 , 00000030
SL     XM 0401 XMLD  ENTRY LOCATE_AND_LOCK_TRANDEF CWXN
SL     XM 0402 XMLD  EXIT  LOCATE_AND_LOCK_TRANDEF/OK 240D79F0 , 0000007F,CWXN
SL     DS 0002 DSAT  ENTRY ATTACH                24ED0650,0,1,NON_SYSTEM,24ED0650 , 0000211C
SL     DS 0003 DSAT  EXIT  ATTACH/OK             0C8C0097
SL     XM 1102 XMAT  EXIT  ATTACH/OK             0000211C
SL     DS 0004 DSSR  ENTRY WAIT_MVS              SODOMAIN,22C2B048,NO,MISC,SO_NOWORK
QR     DS 0012 DSKE  ENTRY TASK_REPLY            23E92080,420B9980
QR     XM 1305 XMTA  ENTRY TASK_REPLY            24ED0650,0C8C0097,0C8C0097
QR     AP 0590 APXM  ENTRY INIT_XM_CLIENT        YES
QR     AP EA00 TMP   ENTRY LOCATE                PFT,DFHCICST
QR     AP EA01 TMP   EXIT  LOCATE                PFT,DFHCICST,22C16B80,NORMAL
QR     AP 0591 APXM  EXIT  INIT_XM_CLIENT/OK
QR     DS 0002 DSAT  ENTRY SET_PRIORITY          1
QR     DS 0003 DSAT  EXIT  SET_PRIORITY/OK
QR     AP 0590 APXM  ENTRY BIND_XM_CLIENT
QR     AP 0591 APXM  EXIT  BIND_XM_CLIENT/OK
QR     AP 0590 APXM  ENTRY RMI_START_OF_TASK
QR     AP 2520 ERM   ENTRY CALL-TRUES-FOR-TASK-START
QR     AP 2521 ERM   EXIT  CALL-TRUES-FOR-TASK-START
QR     AP 0591 APXM  EXIT  RMI_START_OF_TASK/OK
QR     AP 1940 APLI  ENTRY START_PROGRAM         DFHWBXN,NOCEDF,FULLAPI,EXEC,NO,2402BCF0,00000000 , 00000000,1,NO
QR     AP E160 EXEC  ENTRY ADDRESS               AT X'A4FE13E8',NOHANDLE,PLX,31840000
QR     AP E161 EXEC  EXIT  ADDRESS               X'24FB80D0' AT X'A4FE13E8',0,0,NOHANDLE,PLX,31840000
QR     AP E160 EXEC  ENTRY HANDLE                ABEND X'25CF65CE' AT X'A4FE107C',PLX,32480000
QR     AP E161 EXEC  EXIT  HANDLE                ABEND X'25CF65CE' AT X'A4FE107C',0,0,PLX,32480000
QR     DS 0004 DSSR  ENTRY ADD_SUSPEND           WBALIAS,CWBA
QR     DS 0005 DSSR  EXIT  ADD_SUSPEND/OK        00770083
QR     XM 1001 XMIQ  ENTRY INQUIRE_TRANSACTION   24FE1640 , 00000000 , 00000008
QR     XM 1002 XMIQ  EXIT  INQUIRE_TRANSACTION/OK C,24FE1640 , 00000000 , 00000008
SO     DS 0004 DSSR  ENTRY WAIT_MVS              SOCKET,1E,24FA25D0,YES,SECOND,IDLE,RECEIVE
SO     DS 0005 DSSR  EXIT  WAIT_MVS/OK
SO     DS 0004 DSSR  ENTRY WAIT_MVS              SOCKET,24FA25D0,YES,IDLE,RECEIVE
SO     DS 0005 DSSR  EXIT  WAIT_MVS/OK
QR     AP 4800 CCNV  ENTRY VERIFY_IANA_CCSID     utf-8
QR     AP 4801 CCNV  EXIT  VERIFY_IANA_CCSID/OK  4B8
SO     DS 0004 DSSR  ENTRY WAIT_MVS              SOCKET,24FA25D0,YES,IDLE,RECEIVE
SO     DS 0005 DSSR  EXIT  WAIT_MVS/OK
QR     AP 4800 CCNV  ENTRY CONVERT_DATA          333,25,26715000 , 00000000 , 00000093,26715000 , 00000000 , 00000093
QR     AP 4801 CCNV  EXIT  CONVERT_DATA/OK       26715000 , 00000093 , 00000093,26715000 , 00000093 , 00000093
QR     WB 0905 WBUR  DATA  URI-MAPPING-ELEMENT
QR     AP 4800 CCNV  ENTRY CONVERT_DATA          4B8,25,22C25760 , 00000000 , 0000051E,26715093 , 00000000 , 00007F6C
QR     AP 4801 CCNV  EXIT  CONVERT_DATA/OK       22C25760 , 0000051E , 0000051E,26715093 , 0000051E , 00007F6C
QR     XM 1101 XMAT  ENTRY ATTACH                CPIH,NONE,C,YES,YES,WEB,250F1030 , 00000360,SAME
QR     XM 0401 XMLD  ENTRY LOCATE_AND_LOCK_TRANDEF CPIH
QR     XM 0402 XMLD  EXIT  LOCATE_AND_LOCK_TRANDEF/OK 24F46B90 , 000000BA,CPIH
QR     DS 0002 DSAT  ENTRY ATTACH                24ED04C8,0,1,NON_SYSTEM,24ED04C8 , 0000212C
QR     DS 0003 DSAT  EXIT  ATTACH/OK             0C020051
QR     XM 1102 XMAT  EXIT  ATTACH/OK             0000212C
QR     DS 0004 DSSR  ENTRY DELETE_SUSPEND        00770083
QR     DS 0005 DSSR  EXIT  DELETE_SUSPEND/OK
QR     AP 1941 APLI  EXIT  START_PROGRAM/OK      ,NO,DFHWBXN
```

URI Mapping

Of interest in the trace entries for Example 9-10 on page 234, is further detail on the trace point for **WB 0905** that shows the URI mapping in the CICS FULL trace output in Example 9-11. In this entry you can see the following in the eyecatcher area:

- Transaction **CPIH**
- Pipeline **WSPIPE0**1
- The name of the Web service to be invoked - **RetrieveAddress**
- The relative URI for the service - **/cicswsap/RetrieveAddress**

Example 9-11 The FULL trace output showing URI mapping

```
WB 0905 WBUR  DATA - URI-MAPPING-ELEMENT

TASK-00211 KE_NUM-00E3 TCB-QR   /007DCD98 RET-A3014F40 TIME-22:39:59.4649770986 INTERVAL-00.0000037656        =007768=
 1-0000  24FBD5D0 000000F0                                                        *..N}...0                        *
 2-0000  00F06EC4 C6C8E6C2 E4D9C9D4 C1D74040  24FBD6C0 24FBDA80 5BF5F1F1 F2F8F040  *.0>DFHWBURIMAP  ..O{....$511280 *
   0020  24FC10E0 24FC8150 01038330 24FC5330  00000000 00000000 00000019 00000000  *...\..a&..c.....................*
   0040  C3D7C9C8 00000000 40404040 40404040  40404040 40404040 40404040 40404040  *CPIH....                        *
   0060  E6E2D7C9 D7C5F0F1 D985A399 8985A585  C1848499 85A2A240 40404040 40404040  *WSPIPE01RetrieveAddress         *
   0080  40404040 40404040 00000000 00000000  00000000 00000000 00000000 00000000  *        ........................*
   00A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   00C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   00E0  00000015 00000000 00000000 00000000                                      *................                *
 3-0000  00B06EC4 C6C8E6C2 E5C9D9E3 C8D6E2E3  22C25070 24FC1030 24FC8070 00000000  *..>DFHWBVIRTHOST.B&.............*
   0020  80000002 40404040 40404040 00000015  00000000 00015C40 40404040 40404040  *....        ..........*         *
   0040  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
   0060  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
   0080  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
   00A0  40404040 40404040 40400000 00000000                                      *          ......                *
 4-0000  00806EC4 C6C8E6C2 E4D9C9D4 C1D7E7D5  24FBD5D0 D7400019 61838983 A2A6A281  *..>DFHWBURIMAPXN..N}P ../cicswsa*
   0020  9761D985 A3998985 A585C184 849985A2  A2404040 40404040 40404040 40404040  *p/RetrieveAddress               *
```

2. A service provider pipeline is started

Transaction CPIH invokes the CICS program DFHPIDSH—the pipeline HTTP inbound router module—starting a service provider pipeline. This is shown in Example 9-12.

Example 9-12 Service provider pipeline is started

```
QR    AP EA00 TMP   ENTRY LOCATE                PFT,DFHCICST
QR    AP EA01 TMP   EXIT  LOCATE                PFT,DFHCICST,22C16B80,NORMAL
QR    AP 0591 APXM  EXIT  INIT_XM_CLIENT/OK
QR    DS 0002 DSAT  ENTRY SET_PRIORITY          1
QR    DS 0003 DSAT  EXIT  SET_PRIORITY/OK
QR    AP 0590 APXM  ENTRY BIND_XM_CLIENT
QR    AP 0591 APXM  EXIT  BIND_XM_CLIENT/OK
QR    XM 1204 XMER  ENTRY INQUIRE_DEFERRED_MESSAGE
QR    XM 1205 XMER  EXIT  INQUIRE_DEFERRED_MESSAGE/EXCEPTION MESSAGE_NOT_FOUND,
QR    AP 0590 APXM  ENTRY RMI_START_OF_TASK
QR    AP 2520 ERM   ENTRY CALL-TRUES-FOR-TASK-START
QR    AP 2521 ERM   EXIT  CALL-TRUES-FOR-TASK-START
QR    AP 0591 APXM  EXIT  RMI_START_OF_TASK/OK
QR    AP 1940 APLI  ENTRY START PROGRAM         DFHPIDSH,NOCEDF,FULLAPI,EXEC,NO,2402BD34,00000000 , 00000000,1,NO
QR    AP 09D0 PIDSH ENTRY SOAP HTTP INBOUND ROUTER
QR    XM 1001 XMIQ  ENTRY INQUIRE_TRANSACTION
QR    XM 1002 XMIQ  EXIT  INQUIRE_TRANSACTION/OK C
QR    PI 0A20 PIIS  ENTRY INIT
QR    DS 0004 DSSR  ENTRY ADD_SUSPEND           PIISLSTN
QR    DS 0005 DSSR  EXIT  ADD_SUSPEND/OK        00770085
QR    PI 0A21 PIIS  EXIT  INIT                  00000001,00000088
QR    PI 0A2F PIIS  ENTRY INIT NODES
QR    PI 0A34 PIIS  EVENT ADD_NODE              DFHPISN1
QR    PI 0A30 PIIS  EXIT  INIT_NODES            00000001,00000001
QR    PI 0A22 PIIS  ENTRY RUN
QR    WB 0307 WBAP  DATA  READ_HEADER
QR    WB 0307 WBAP  DATA  READ_HEADER
QR    PI 0A28 PIIS  EVENT STATE                 F,T
QR    PI 0A2A PIIS  EVENT INITIAL_STATE         T
QR    PI 0A31 PIIS  EVENT REQUEST_CNT
QR    PI 0A40 PIIS  EVENT FUNCTION CNT          PROCESS-REQUEST
QR    PI 0A31 PIIS  EVENT REQUEST_CNT
QR    AP 4800 CCNV  ENTRY CREATE_CONVERSION_TOKEN 25,4B8
QR    AP 4801 CCNV  EXIT  CREATE_CONVERSION_TOKEN/OK 2601C11C , 00000000
```

Pipeline details

The FULL trace entry for trace point ID **PI 0A2F** shows the details of the PIPELINE, including the configuration file name, the HFS shelf, and pickup directories. This is seen in Example 9-13 and shows the following:

– The pipeline name - WSPIPE01

– The configuration file name - **`/usr/lpp/cicsts/samples/pipelines/basicsoap11provider.xml`**

– The HFS shelf directory - **`u/jnott/cicswsap/shelf`**

– The HFS pickup directory - **`u/jnott/cicswsap/wsbind/provider`**

Example 9-13 Full trace entry showing pipeline details

```
PI 0A2F PIIS ENTRY - INIT_NODES -

TASK-00212 KE_NUM-00E4 TCB-QR   /007DCD98 RET-A59FDBA2 TIME-22:39:59.4732479272 INTERVAL-00.0008575942        =007997=
 1-0000  6EC4C6C8 D7C9C9E2 40404040 40404040  00000000 00000000 22C41064 22C41064  *>DFHPIIS        .........D...D..*
   0020  E6E2D7C9 D7C5F0F1 D985A399 8985A585  C1848499 85A2A240 40404040 40404040  *WSPIPE01RetrieveAddress         *
   0040  40404040 40404040 00000000 00000000  00000000 40404040 40404040 01000015  *        ............        ....*
   0060  00000000 00000000 00000000 00000000  D7C6D5D5 D5D5D5D5 D5D5D5D5 D5404040  *................PFNNNNNNNNNNN   *
   0080  00000000 00000000 40404040 40404040  00000000 00000000 00000000 00000000  *........        ................*
   00A0  6EC4C6C8 D7C9C9E2 6DD5D6C4 C5404040  40404040 40404040 00000000 00000000  *>DFHPIIS_NODE           ........*
   00C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   00E0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   0100  24EEA128 24EEA128 00000000 00000000  24EEA138 24EEA138 0000000A A385A7A3  *............................text*
   0120  61A79493 40404040 40404040 40404040  40404040 40404040 40404040 40404040  */xml                            *
   0140  40404040 40404040 40404040 40404040  40404040 40404040 40404040 00000000  *                            ....*
   0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000000C8  *...............................H*
   0180  D6D24040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *OK                              *
   01A0  C8E6E2C1 D7D7D6D9 E3404040 40404040  40404040 40404040 40404040 40404040  *HWSAPPORT                       *
   ......
 2-0000  04646EC4 C6C8D7C9 D7C5C240 40404040  00000000 00000000 24F4B4B0 22C41608  *..>DFHPIPEB     .........4...D..*
   0020  E6E2D7C9 D7C5F0F1 00000001 0000001B  00000000 00000000 00000000 00000000  *WSPIPE01........................*
   ......
   0120  22C415D0 00000000 00000000 E6E2D7C9  D7C5F0F1 22B8EAA8 00000000 00000000  *.D.}........WSPIPE01...y........*
   0140  C4C6C8D7 C9E3D740 01010261 A4A29961  93979761 838983A2 A3A26183 8983A2A3  *DFHPITP .../usr/lpp/cicsts/cicst*
   0160  A2F3F161 A2819497 9385A261 97899785  93899585 A2618281 A28983A2 968197F1  *s31/samples/pipelines/basicsoap1*
   0180  F1979996 A5898485 994BA794 93000000  00000000 00000000 00000000 00000000  *1provider.xml...................*
   01A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   ......
   0240  00000000 00000000 00000061 A4619195  96A3A361 838983A2 A6A28197 61A28885  *.........../u/jnott/cicswsap/she*
   0260  93866100 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *lf/.............................*
   ......
   0340  00000000 00000000 00000061 A4619195  96A3A361 838983A2 A6A28197 61A6A282  *.........../u/jnott/cicswsap/wsb*
   0360  89958461 979996A5 89848599 61000000  00000000 00000000 00000000 00000000  *ind/provider/...................*
   0380  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   03A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
   03C0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
```

The incoming request

The trace point **PI 0A41** contains the incoming request. The FULL entry showing the entire request is shown in Example 9-14.

Example 9-14 Full trace record of the incoming request data

```
PI 0A31 PIIS EVENT - REQUEST_CNT -

TASK-00212 KE_NUM-00E4 TCB-QR   /007DCD98 RET-A59FDBA2 TIME-22:39:59.4787713803 INTERVAL-00.0000002968        =008117=
  1-0000  3C3F786D 6C207665 7273696F 6E3D2231  2E30223F 3E3C746E 733A456E 76656C6F  *<?xml version="1.0"?><tns:Envelo*
    0020  70652078 6D6C6E73 3A746E73 3D226874  74703A2F 2F736368 656D6173 2E786D6C  *pe xmlns:tns="http://schemas.xml*
    0040  736F6170 2E6F7267 2F736F61 702F656E  76656C6F 70652F22 20786D6C 6E733A72  *soap.org/soap/envelope/" xmlns:r*
    0060  65716E73 3D226874 74703A2F 2F777777  2E495453 4F474830 332E4954 534F4748  *eqns="http://www.ITSOGH03.ITSOGH*
    0080  43492E52 65717565 73742E63 6F6D2220  786D6C6E 733A7265 736E733D 22687474  *CI.Request.com" xmlns:resns="htt*
    00A0  703A2F2F 7777772E 4954534F 47483033  2E495453 4F474843 492E5265 73706F6E  *p://www.ITSOGH03.ITSOGHCI.Respon*
    00C0  73652E63 6F6D2220 786D6C6E 733A4E53  313D2268 7474703A 2F2F7777 772E4954  *se.com" xmlns:NS1="http://www.IT*
    00E0  534F5241 30332E49 54534F52 4143412E  52657175 6573742E 636F6D22 20786D6C  *SORA03.ITSORACA.Request.com" xml*
    0100  6E733A4E 53323D22 68747470 3A2F2F77  77772E49 54534F52 4130332E 4954534F  *ns:NS2="http://www.ITSORA03.ITSO*
    0120  52414341 2E526573 706F6E73 652E636F  6D222078 6D6C6E73 3A4E5333 3D226874  *RACA.Response.com" xmlns:NS3="ht*
    0140  74703A2F 2F777777 2E495453 4F534130  332E4954 534F5341 43412E52 65717565  *tp://www.ITSOSA03.ITSOSACA.Reque*
    0160  73742E63 6F6D2220 786D6C6E 733A4E53  343D2268 7474703A 2F2F7777 772E4954  *st.com" xmlns:NS4="http://www.IT*
    0180  534F5341 30332E49 54534F53 4143412E  52657370 6F6E7365 2E636F6D 2220786D  *SOSA03.ITSOSACA.Response.com" xm*
    01A0  6C6E733A 4E53353D 22687474 703A2F2F  7777772E 4954534F 534E3033 2E495453  *lns:NS5="http://www.ITSOSN03.ITS*
    01C0  4F534E43 412E5265 71756573 742E636F  6D222078 6D6C6E73 3A4E5336 3D226874  *OSNCA.Request.com" xmlns:NS6="ht*
    01E0  74703A2F 2F777777 2E495453 4F534E30  332E4954 534F534E 43412E52 6573706F  *tp://www.ITSOSN03.ITSOSNCA.Respo*
    0200  6E73652E 636F6D22 20786D6C 6E733A78  73643D22 68747470 3A2F2F77 77772E77  *nse.com" xmlns:xsd="http://www.w*
    0220  332E6F72 672F3230 30312F58 4D4C5363  68656D61 2220786D 6C6E733A 6768746E  *3.org/2001/XMLSchema" xmlns:ghtn*
    0240  733D2268 7474703A 2F2F7777 772E4954  534F4748 30332E49 54534F47 4843492E  *s="http://www.ITSOGH03.ITSOGHCI.*
    0260  636F6D22 20786D6C 6E733A4E 53373D22  68747470 3A2F2F77 77772E49 54534F52  *com" xmlns:NS7="http://www.ITSOR*
    0280  4130332E 4954534F 52414341 2E636F6D  2220786D 6C6E733A 7361746E 733D2268  *A03.ITSORACA.com" xmlns:satns="h*
    02A0  7474703A 2F2F7777 772E4954 534F5341  30332E49 54534F53 4143412E 636F6D22  *ttp://www.ITSOSA03.ITSOSACA.com"*
    02C0  20786D6C 6E733A73 6E746E73 3D226874  74703A2F 2F777777 2E495453 4F534E30  * xmlns:sntns="http://www.ITSOSN0*
    02E0  332E4954 534F534E 43412E63 6F6D2220  786D6C6E 733A7873 693D2268 7474703A  *3.ITSOSNCA.com" xmlns:xsi="http:*
    0300  2F2F7777 772E7733 2E6F7267 2F323030  312F584D 4C536368 656D612D 696E7374  *//www.w3.org/2001/XMLSchema-inst*
    0320  616E6365 223E3C74 6E733A42 6F64793E  3C4E5331 3A495453 4F524130 334F7065  *ance"><tns:Body><NS1:ITSORA03Ope*
    0340  72617469 6F6E3E3C 4E53313A 72616361  3E3C4E53 313A4669 7273744E 616D653E  *ration><NS1:raca><NS1:FirstName>*
    0360  54686F6D 61733C2F 4E53313A 46697273  744E616D 653E3C4E 53313A4D 6964646C  *Thomas</NS1:FirstName><NS1:Middl*
    0380  654E616D 653E4A3C 2F4E5331 3A4D6964  646C654E 616D653E 3C4E5331 3A4C6173  *eName>J</NS1:MiddleName><NS1:Las*
    03A0  744E616D 653E5761 74736F6E 3C2F4E53  313A4C61 73744E61 6D653E3C 4E53313A  *tName>Watson</NS1:LastName><NS1:*
    03C0  41646472 65737348 6173683E 31333837  36363236 36393C2F 4E53313A 41646472  *AddressHash>1387662669</NS1:Addr*
    03E0  65737348 6173683E 3C4E5331 3A436C69  656E7449 643E313C 2F4E5331 3A436C69  *essHash><NS1:ClientId>1</NS1:Cli*
    0400  656E7449 643E3C4E 53313A41 64647265  73734C69 6E65313E 6E756C6C 3C2F4E53  *entId><NS1:AddressLine1>null</NS*
    0420  313A4164 64726573 734C696E 65313E3C  4E53313A 41646472 6573734C 696E6532  *1:AddressLine1><NS1:AddressLine2*
    0440  3E6E756C 6C3C2F4E 53313A41 64647265  73734C69 6E65323E 3C4E5331 3A537562  *>null</NS1:AddressLine2><NS1:Sub*
    0460  7572623E 6E756C6C 3C2F4E53 313A5375  62757262 3E3C4E53 313A5374 6174653E  *urb>null</NS1:Suburb><NS1:State>*
    0480  6E756C6C 3C2F4E53 313A5374 6174653E  3C4E5331 3A506F73 74636F64 653E6E75  *null</NS1:State><NS1:Postcode>nu*
    04A0  6C6C3C2F 4E53313A 506F7374 636F6465  3E3C4E53 313A7263 3E2D313C 2F4E5331  *ll</NS1:Postcode><NS1:rc>-1</NS1*
    04C0  3A72633E 3C4E5331 3A726561 736F6E3E  6E756C6C 3C2F4E53 313A7265 61736F6E  *:rc><NS1:reason>null</NS1:reason*
    04E0  3E3C2F4E 53313A72 6163613E 3C2F4E53  313A4954 534F5241 30334F70 65726174  *></NS1:raca></NS1:ITSORA03Operat*
    0500  696F6E3E 3C2F746E 733A426F 64793E3C  2F746E73 3A456E76 656C6F70 653E      *ion></tns:Body></tns:Envelope>  *
```

3. Handling the Request

The trace in Example 9-15 shows the CICS processing that continues:

– Program DFHPISN1 starts, which is the handler program for SOAP 1.1.

– DFHPIEP is invoked, which parses the incoming SOAP request.

Example 9-15 Parsing the incoming SOAP request

```
QR    AP 1940 APLI  ENTRY START_PROGRAM       DFHPISN1,NOCEDF,FULLAPI,EXEC,NO,2402BD78,00000000 , 00000000,2,NO
L800B PI 0C15 PISN  ENTRY SOAP_PARSER
L800B AP 1940 APLI  ENTRY START_PROGRAM       DFHPIEP,NOCEDF,FULLAPI,EXEC,NO,2402BDBC,23EA0730 , 00000010,3,NO
L800B AP 1948 APLI  EVENT CALL-TO-LE/370      Thread_Initialization DFHPIEP
L800B AP 1949 APLI  EVENT RETURN-FROM-LE/370  Thread_Initialization OK DFHPIEP
L800B AP 1948 APLI  EVENT CALL-TO-LE/370      Rununit_Init_&_Begin_Invo DFHPIEP
L800B AP E160 EXEC  ENTRY ADDRESS             AT X'A6E01A30',SYSEIB,ASM,00000229
L800B AP E161 EXEC  EXIT  ADDRESS             X'22CB2494' AT X'A6E01A30',0,0,SYSEIB,ASM,00000229
L800B AP E160 EXEC  ENTRY GETMAIN             AT X'26E0196C',514 AT X'A6E0195C',USERDATAKEY,SYSEIB,NOHANDLE,ASM,000
L800B AP E161 EXEC  EXIT  GETMAIN             X'26E12B78' AT X'26E0196C',514 AT X'A6E0195C',USERDATAKEY,0,0,SYSEIB
L800B AP E160 EXEC  ENTRY ADDRESS             AT X'26E00050',AT X'A6E0AB90',NOHANDLE,PL/I,184.1
L800B AP E161 EXEC  EXIT  ADDRESS             X'26E000D0' AT X'26E00050',X'23EA0730' AT X'A6E0AB90',0,0,NOHANDLE,PL
L800B AP E160 EXEC  ENTRY GETMAIN             AT X'26E00578',4080 AT X'A6E00568',USERDATAKEY,SYSEIB,NOHANDLE,ASM,00
L800B AP E161 EXEC  EXIT  GETMAIN             X'26E12D98' AT X'26E00578',4080 AT X'A6E00568',USERDATAKEY,0,0,SYSEIB
L800B AP E160 EXEC  ENTRY GETMAIN             AT X'26E00578',4080 AT X'A6E00568',USERDATAKEY,SYSEIB,NOHANDLE,ASM,00
L800B AP E161 EXEC  EXIT  GETMAIN             X'26E13D98' AT X'26E00578',4080 AT X'A6E00568',USERDATAKEY,0,0,SYSEIB
L800B AP 1949 APLI  EVENT RETURN-FROM-LE/370  Rununit_Init_&_Begin_Invo OK DFHPIEP
L800B AP 1948 APLI  EVENT CALL-TO-LE/370      Rununit_End_Invocation DFHPIEP
L800B AP 1949 APLI  EVENT RETURN-FROM-LE/370  Rununit_End_Invocation OK DFHPIEP
L800B AP 1948 APLI  EVENT CALL-TO-LE/370      Rununit_Termination DFHPIEP
L800B AP E160 EXEC  ENTRY FREEMAIN            AT X'A6E13D98',SYSEIB,NOHANDLE,ASM,00000404
L800B AP E161 EXEC  EXIT  FREEMAIN            AT X'A6E13D98',0,0,SYSEIB,NOHANDLE,ASM,00000404
L800B AP E160 EXEC  ENTRY FREEMAIN            AT X'A6E12D98',SYSEIB,NOHANDLE,ASM,00000404
L800B AP E161 EXEC  EXIT  FREEMAIN            AT X'A6E12D98',0,0,SYSEIB,NOHANDLE,ASM,00000404
L800B AP 1949 APLI  EVENT RETURN-FROM-LE/370  Rununit_Termination OK DFHPIEP
L800B AP 1941 APLI  EXIT  START_PROGRAM/OK    ,NO,DFHPIEP
L800B PI 0C16 PISN  EXIT  SOAP_PARSER         0,0,0
L800B PI 0C17 PISN  ENTRY CALL_HEADERS
L800B PI 0C82 PISH  DATA  PROCESS-REQUEST     24FE15B0,FF00000024FE107D0000050924FE107E000000030000000824FE108224FE
L800B PI 0C82 PISH  DATA  PROCESS-REQUEST     24FE15B0,FF00000024FE107D0000050924FE107E000000030000000824FE108224FE
L800B PI 0C18 PISN  EXIT  CALL_HEADERS        1,0
```

4. CICS SOAP application driver

CICS now invokes program DFHPITP, which processes the incoming SOAP message. The trace in Example 9-16 clearly shows the elements of the message that are to be processed by the back end logic in ITSORA03.

Example 9-16 Soap elements for the Web service

```
L800B AP 1940 APLI  ENTRY START PROGRAM         DFHPITP,NOCEDF,FULLAPI,EXEC,NO,2402BE00,00000000 , 00000000,3,NO
L800B AP 09F5 PITP  ENTRY PROCESS_SOAP_REQUEST
L800B PI 1012 PITL  DATA  WSBIND_TIMESTAMP      -200702010011
L800B AP 4800 CCNV  ENTRY CREATE_CONVERSION_TOKEN 4B8,25
L800B AP 4801 CCNV  EXIT  CREATE_CONVERSION_TOKEN/OK 2601C170 , 00000000
L800B AP 4800 CCNV  ENTRY CONVERT_DATA          4B8,25,24FDDAA8 , 00000000 , 000001E9,26E05A28 , 00000000 , 000007A4
L800B AP 4801 CCNV  EXIT  CONVERT_DATA/OK       24FDDAA8 , 000001E9 , 000001E9,26E05A28 , 000001E9 , 000007A4,2601C17
L800B AP 4800 CCNV  ENTRY CREATE_CONVERSION_TOKEN 4B8,25
L800B AP 4801 CCNV  EXIT  CREATE_CONVERSION_TOKEN/OK 2601C170 , 00000000
L800B AP 4800 CCNV  ENTRY CONVERT_DATA          4B8,25,24FDD778 , 00000000 , 00000303,26E061E8 , 00000000 , 00000C0C
L800B AP 4801 CCNV  EXIT  CONVERT_DATA/OK       24FDD778 , 00000303 , 00000303,26E061E8 , 00000303 , 00000C0C,2601C17
L800B PI 0F36 PICC  DATA  INBOUND SOAP BODY
L800B AP 4800 CCNV  ENTRY CONVERT_DATA          4B8,25,22F08068 , 00000000 , 00000003,23EE6980 , 00000000 , 00000003
L800B AP 4801 CCNV  EXIT  CONVERT_DATA/OK       22F08068 , 00000003 , 00000003,23EE6980 , 00000003 , 00000003
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    tns:Body
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:ITSORA03Operation
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:raca
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:FirstName
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:FirstName
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:MiddleName
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:MiddleName
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:LastName
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:LastName
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:AddressHash
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:AddressHash
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:ClientId
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:ClientId
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:AddressLine1
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:AddressLine1
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:AddressLine2
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:AddressLine2
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:Suburb
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:Suburb
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:State
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:State
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:Postcode
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:Postcode
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:rc
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:rc
L800B PI 0F3E PICC  DATA  SOAP_ELEMENT_START    NS1:reason
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:reason
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:raca
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      NS1:ITSORA03Operation
L800B PI 0F3F PICC  DATA  SOAP_ELEMENT_END      tns:Body
L800B PI 0F37 PICC  DATA  OUTPUT_COMMAREA_DATA
```

The Inbound SOAP body

The FULL trace entry in Example 9-17 shows that the inbound SOAP body is displayed in the eyecatcher area of trace point ID **PI 0F36.**

Example 9-17 Trace of the inbound SOAP body

```
PI 0F36 PICC  DATA  - INBOUND_SOAP_BODY

 TASK-00212 KE_NUM-00E4 TCB-L800B/00799270 RET-A2F1F944 TIME-22:39:59.4887448779 INTERVAL-00.0008551230        =008342=
  1-0000  4CA395A2 7AC29684 A86E4CD5 E2F17AC9  E3E2D6D9 C1F0F3D6 97859981 A3899695  *<tns:Body><NS1:ITSORA03Operation*
    0020  6E4CD5E2 F17A9981 83816E4C D5E2F17A  C68999A2 A3D58194 856EE388 969481A2  *><NS1:raca><NS1:FirstName>Thomas*
    0040  4C61D5E2 F17AC689 99A2A3D5 8194856E  4CD5E2F1 7AD48984 849385D5 8194856E  *</NS1:FirstName><NS1:MiddleName>*
    0060  D14C61D5 E2F17AD4 89848493 85D58194  856E4CD5 E2F17AD3 81A2A3D5 8194856E  *J</NS1:MiddleName><NS1:LastName>*
    0080  E681A3A2 96954C61 D5E2F17A D381A2A3  D5819485 6E4CD5E2 F17AC184 849985A2  *Watson</NS1:LastName><NS1:Addres*
    00A0  A2C881A2 886EF1F3 F8F7F6F6 F2F6F6F9  4C61D5E2 F17AC184 849985A2 A2C881A2  *sHash>1387662669</NS1:AddressHas*
    00C0  886E4CD5 E2F17AC3 93898595 A3C9846E  F14C61D5 E2F17AC3 93898595 A3C9846E  *h><NS1:ClientId>1</NS1:ClientId>*
    00E0  4CD5E2F1 7AC18484 9985A2A2 D3899585  F16E95A4 93934C61 D5E2F17A C1848499  *<NS1:AddressLine1>null</NS1:Addr*
    0100  85A2A2D3 899585F1 6E4CD5E2 F17AC184  849985A2 A2D38995 85F26E95 A493934C  *essLine1><NS1:AddressLine2>null<*
    0120  61D5E2F1 7AC18484 9985A2A2 D3899585  F26E4CD5 E2F17AE2 A482A499 826E95A4  */NS1:AddressLine2><NS1:Suburb>nu*
    0140  93934C61 D5E2F17A E2A482A4 99826E4C  D5E2F17A E2A381A3 856E95A4 93934C61  *ll</NS1:Suburb><NS1:State>null</*
    0160  D5E2F17A E2A381A3 856E4CD5 E2F17AD7  96A2A383 9684856E 95A49393 4C61D5E2  *NS1:State><NS1:Postcode>null</NS*
    0180  F17AD796 A2A38396 84856E4C D5E2F17A  99836E60 F14C61D5 E2F17A99 836E4CD5  *1:Postcode><NS1:rc>-1</NS1:rc><N*
    01A0  E2F17A99 8581A296 956E95A4 93934C61  D5E2F17A 998581A2 96956E4C 61D5E2F1  *S1:reason>null</NS1:reason></NS1*
    01C0  7A998183 816E4C61 D5E2F17A C9E3E2D6  D9C1F0F3 D6978599 81A38996 956E4C61  *:raca></NS1:ITSORA03Operation></*
    01E0  A395A27A C29684A8 6E                                                      *tns:Body>                       *
```

5. Invoke the back end program

Our service provider business logic is invoked. The trace entries in Example 9-18 show that the following actions are performed:

- Business logic ITSORA03 is started.
- Calls to DB2 to see if there is an entry for this client to perform this query for new address details, to retrieve the address details, and to insert an audit record.

Example 9-18 Invoking the service provider program ITSORA043

```
L800B AP 19A0 APLX  ENTRY START_PROGRAM        ITSORA03,CEDF,FULLAPI,EXEC,NO,2402BE44,26E06E08 , 000001DC,4,NO
X9000 DS 0010 DSBR  ENTRY INQUIRE_TCB
X9000 DS 0011 DSBR  EXIT  INQUIRE_TCB/OK       22C16D40
X9000 AP E160 EXEC  ENTRY ADDRESS              AT X'A6714A30',NOHANDLE,C/370,00026300
X9000 AP E161 EXEC  EXIT  ADDRESS              X'26E06E08' AT X'A6714A30',0,0,NOHANDLE,C/370,00026300
X9000 AP E160 EXEC  ENTRY WRITEQ               TS 'RA03    ' AT X'26712D64','Thomas
X9000 AP E161 EXEC  EXIT  WRITEQ               TS 'RA03    ' AT X'26712D64','Thomas
X9000 AP 2520 ERM   ENTRY C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
L800B AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT
L800B AP 3250 D2D2  ENTRY DB2_API_CALL         24039330
L800B AP 3251 D2D2  EXIT  DB2_API_CALL/OK
L800B AP 3181 D2EX1 EXIT  APPLICATION-REQUEST  SQLCODE 0 RETURNED ON EXEC SQL SELECT
X9000 AP 2521 ERM   EXIT  C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
X9000 AP 2520 ERM   ENTRY C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
L800B AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT
L800B AP 3250 D2D2  ENTRY DB2_API_CALL         24039330
L800B AP 3251 D2D2  EXIT  DB2_API_CALL/OK
L800B AP 3181 D2EX1 EXIT  APPLICATION-REQUEST  SQLCODE 0 RETURNED ON EXEC SQL SELECT
X9000 AP 2521 ERM   EXIT  C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
X9000 AP 2520 ERM   ENTRY C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
L800B AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL INSERT
L800B AP 3250 D2D2  ENTRY DB2_API_CALL         24039330
L800B AP 3251 D2D2  EXIT  DB2_API_CALL/OK
L800B AP 3181 D2EX1 EXIT  APPLICATION-REQUEST  SQLCODE 0 RETURNED ON EXEC SQL INSERT
X9000 AP 2521 ERM   EXIT  C-APPLICATION-CALL-TO-TRUE(DSNCSQL )
X9000 AP E160 EXEC  ENTRY RETURN               NOHANDLE C/370 00061200
X9000 DS 0010 DSBR  ENTRY INQUIRE_TASK
X9000 DS 0011 DSBR  EXIT  INQUIRE_TASK/OK      24EAE030 , 00000003
```

6. Sending a response to the client

CICS now builds a response and sends it back to the client using the same pipeline. The trace entries in Example 9-19 show the response being sent.

Example 9-19 Sending a response to the client

```
L800B PI 0F06 PIII  DATA  OUTBOUND_COMMAREA_DATA
L800B PI 0F07 PIII  DATA  OUTBOUND_SOAP_BODY
L800B AP 4800 CCNV  ENTRY CREATE_CONVERSION_TOKEN 25,4B8
L800B AP 4801 CCNV  EXIT  CREATE_CONVERSION_TOKEN/OK 2601C11C , 00000000
L800B AP 4800 CCNV  ENTRY CONVERT_DATA          25,4B8,26E06FF8 , 00000000 , 00000376,250F3028 , 00000000 , 00000DD8
L800B AP 4801 CCNV  EXIT  CONVERT_DATA/OK       26E06FF8 , 00000376 , 00000376,250F3028 , 00000376 , 00000DD8,2601C11
L800B AP 09F6 PITP  EXIT  PROCESS_SOAP_REQUEST
L800B AP E160 EXEC  ENTRY RETURN                PLX 84500000
L800B AP 1941 APLI  EXIT  START_PROGRAM/OK      ,NO,DFHPITP
L800B PI 0C17 PISN  ENTRY CALL_HEADERS
L800B PI 0C82 PISH  DATA  SEND-RESPONSE         FF000000,,,
L800B PI 0C82 PISH  DATA  SEND-RESPONSE         FF000000,,,
L800B PI 0C18 PISN  EXIT  CALL_HEADERS          1,0
L800B AP 1941 APLI  EXIT  START_PROGRAM/OK      ,NO,DFHPISN1
L800B PI 0A28 PIIS  EVENT STATE                 T,B
L800B PI 0A2B PIIS  EVENT FINAL_STATE           B
L800B PI 0A40 PIIS  EVENT FUNCTION_CNT          SEND-RESPONSE
L800B PI 0A32 PIIS  EVENT RESPONSE_CNT
L800B AP 4800 CCNV  ENTRY VERIFY_IANA_CCSID     UTF-8
L800B AP 4801 CCNV  EXIT  VERIFY_IANA_CCSID/OK  4B8
L800B AP 4800 CCNV  ENTRY CONVERT_DATA          25,333,23E9FCB0 , 00000000 , 000000C9,23E9FCB0 , 00000000 , 000000C9
L800B AP 4801 CCNV  EXIT  CONVERT_DATA/OK       23E9FCB0 , 000000C9 , 000000C9,23E9FCB0 , 000000C9 , 000000C9
SO    DS 0004 DSSR  ENTRY WAIT_MVS              SOCKET,24FA25A8,YES,IDLE,SEND
SO    DS 0005 DSSR  EXIT  WAIT_MVS/OK
SO    DS 0004 DSSR  ENTRY WAIT_MVS              SOCKET,24FA25A8,YES,IDLE,SEND
SO    DS 0005 DSSR  EXIT  WAIT_MVS/OK
L800B PI 0A23 PIIS  EXIT  RUN                   00000001,00000001
L800B DS 0004 DSSR  ENTRY DELETE_SUSPEND        00770085
L800B DS 0005 DSSR  EXIT  DELETE_SUSPEND/OK
L800B AP 09D1 PIDSH EXIT  SOAP_HTTP_INBOUND_ROUTER
L800B AP E160 EXEC  ENTRY RETURN                PLX 80500000
L800B AP 1941 APLI  EXIT  START_PROGRAM/OK      ,NO,DFHPIDSH
```

The Outbound SOAP body

The FULL trace entry for trace point ID **PI 0F07** shows the outbound SOAP body that is to be passed back to the client. This is seen in Example 9-20.

Example 9-20   The Outbound SOAP body

```
PI 0F07 PIII  DATA  - OUTBOUND_SOAP_BODY

 TASK-00212 KE_NUM-00E4 TCB-L800B/00799270 RET-A2F20830 TIME-22:39:59.4970271616 INTERVAL-00.0008078750        =008528=
  1-0000  4CE2D6C1 D760C5D5 E57AC296 84A86E4C  C9E3E2D6 D9C1F0F3 D6978599 81A38996  *<SOAP-ENV:Body><ITSORA03Operatio*
    0020  95D985A2 979695A2 8540A794 9395A27E  7F88A3A3 977A6161 A6A6A64B C9E3E2D6  *nResponse xmlns="http://www.ITSO*
    0040  D9C1F0F3 4BC9E3E2 D6D9C1C3 C14BD985  A2979695 A2854B83 96947F6E 4C998183  *RA03.ITSORACA.Response.com"><rac*
    0060  816E4CC6 8999A2A3 D5819485 6EE38896  9481A240 40404040 40404040 40404040  *a><FirstName>Thomas             *
    0080  40404040 40404040 40404040 40404040  40404040 40404040 40404040 4040404C  *                               <*
    00A0  61C68999 A2A3D581 94856E4C D4898484  9385D581 94856ED1 40404040 40404040  */FirstName><MiddleName>J        *
    00C0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
    00E0  40404040 40404040 404C61D4 89848493  85D58194 856E4CD3 81A2A3D5 8194856E  *         </MiddleName><LastName>*
    0100  E681A3A2 96954040 40404040 40404040  40404040 40404040 40404040 40404040  *Watson                          *
    0120  40404040 40404040 40404040 40404040  40404C61 D381A2A3 D5819485 6E4CC184  *                  </LastName><Ad*
    0140  849985A2 A2D38995 85F16EC9 C2D440C7  85969987 89814040 40404040 40404040  *dressLine1>IBM Georgia          *
    0160  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
    0180  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
    01A0  40404040 40404040 40404040 4040404C  61C18484 9985A2A2 D3899585 F16E4CC1  *               </AddressLine1><A*
    01C0  84849985 A2A2D389 9585F26E F140C199  87969581 A4A340E2 A3404040 40404040  *ddressLine2>1 Argonaut St       *
    01E0  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
    0200  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                                *
    0220  40404040 40404040 40404040 40404040  4C61C184 849985A2 A2D38995 85F26E4C  *                </AddressLine2><*
    0240  E2A482A4 99826EE3 828993A2 89404040  40404040 40404040 40404040 40404040  *Suburb>Tbilsi                   *
    0260  40404040 40404040 40404040 40404040  40404040 40404040 404C61E2 A482A499  *                         </Subur*
    0280  826E4CE2 A381A385 6EC7C5D6 D9404040  4040404C 61E2A381 A3856E4C D796A2A3  *b><State>GEOR      </State><Post*
    02A0  83968485 6EF1F1F1 F1F14040 4040404C  61D796A2 A3839684 856E4CC1 84849985  *code>11111     </Postcode><Addre*
    02C0  A2A2C881 A2886EF5 F2F7F8F1 F7F8F8F9  4C61C184 849985A2 A2C881A2 886E4CC3  *ssHash>527817889</AddressHash><C*
    02E0  93898595 A3C9846E F14C61C3 93898595  A3C9846E 4CD58194 85D98586 6EF04C61  *lientId>1</ClientId><NameRef>0</*
    0300  D5819485 D985866E 4C99836E F04C6199  836E4C99 8581A296 956ED6D2 6B409585  *NameRef><rc>0</rc><reason>OK, ne*
    0320  A6408184 849985A2 A2408485 A3818993  A240A2A4 97979389 85844C61 998581A2  *w address details supplied</reas*
    0340  96956E4C 61998183 816E4C61 C9E3E2D6  D9C1F0F3 D6978599 81A38996 95D985A2  *on></raca></ITSORA03OperationRes*
    0360  979695A2 856E4C61 E2D6C1D7 60C5D5E5  7AC29684 A86E                        *ponse></SOAP-ENV:Body>          *
```

The CICS task is then terminated.

9.2.5 Return to the Client Broker on Windows

Having successfully called the CICS RetrieveAddress Web service, the RetrieveAddressWithDB message flow resumes.

Example 9-21   RetrieveAddressWithDB MsgFlow Pt 2

```
Notice there is precious little trace output from the HTTPRequest node. This trace was take at DEBUG (max) level for a
user trace. More interesting information is returned in the HTTPResponse headers. The following is taken from a trace node
entry in the error log:

(0x01000000):HTTPResponseHeader = (
    (0x03000000):X-Original-HTTP-Status-Line = 'HTTP/1.0 200 OK'
    (0x03000000):X-Original-HTTP-Status-Code = 200
    (0x03000000):Server                      = 'IBM_CICS_Transaction_Server/3.1.0(zOS)'
    (0x03000000):Date                        = 'Thu, 22 Feb 2007 03:39:59 GMT'
    (0x03000000):Content-Length              = '00001755'
    (0x03000000):Content-Type                = 'text/xml; charset="UTF-8"'
  )

This says nothing about the success or otherwise of the actuall w-s call, merely that CICS managed to invoke the w-s call
OK.
-------------------------------------------------------------------------------------------------
```

```
2007-02-22 14:40:01.664801    10140   UserTrace   BIP4007I: Message propagated to 'out' terminal of node
'RetrieveAddressWithDB.Retrieve Address'.
2007-02-22 14:40:01.664942    10140   UserTrace   BIP6060I: Parser type ''Properties'' created on behalf of node
'RetrieveAddressWithDB.Retrieve Address' to handle portion of incoming message of length 0 bytes beginning at offset '0'.
2007-02-22 14:40:01.665199    10140   UserTrace   BIP6061I: Parser type ''HTTPResponseHeader'' created on behalf of node
'RetrieveAddressWithDB.Retrieve Address' to handle portion of incoming message of length '201' bytes beginning at offset
'0'. Parser type selected based on value ''WSRSPHDR'' from previous parser.
2007-02-22 14:40:01.665554    10140   UserTrace   BIP6061I: Parser type ''MRM'' created on behalf of node
'RetrieveAddressWithDB.Retrieve Address' to handle portion of incoming message of length '1755' bytes beginning at offset
'201'. Parser type selected based on value ''MRM'' from previous parser.

----------------------------------------------------------------------------------------------
The next node: "Check RC" for copies the input tree to the output tree then  checks the results of the RetrieveAddress w-s
call and sets destination label to "RetrieveOK". Most of these trace entries document the copying of the message tree.
----------------------------------------------------------------------------------------------

2007-02-22 14:40:01.665642    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''DECLARE tns NAMESPACE 'http://schemas.xmlsoap.org/soap/envelope/';'' at ('.tns', '1.1').
2007-02-22 14:40:01.665694    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''DECLARE resns NAMESPACE 'http://www.ITSORA03.ITSORACA.Response.com';'' at ('.resns', '1.1').
2007-02-22 14:40:01.665755    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''DECLARE reqns NAMESPACE 'http://www.ITSORA03.ITSORACA.Request.com';'' at ('.reqns', '1.1').
2007-02-22 14:40:01.665799    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''BEGIN ... END;'' at ('.RetrieveAddressWithDB_CheckRC.Main', '2.2').
2007-02-22 14:40:01.665843    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''CopyMessageHeaders();'' at ('.RetrieveAddressWithDB_CheckRC.Main', '3.3').

----------------------------------------------------------------------------------------------
Lots of trace entries assocated with copying headers edited out here for brevity.
----------------------------------------------------------------------------------------------
RC': Executing statement   ''CopyEntireMessage();'' at ('.RetrieveAddressWithDB_CheckRC.Main', '4.3').

----------------------------------------------------------------------------------------------
Lots of trace entries assocated with copying message edited out here for brevity.
----------------------------------------------------------------------------------------------

2007-02-22 14:40:01.668341    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''IF InputRoot.MRM.resns:raca.resns:rc > 0 THEN... ELSE... END IF;'' at ('.RetrieveAddressWithDB_CheckRC.Main', '6.3').
2007-02-22 14:40:01.669765    10140   UserTrace   BIP2543I: Node 'RetrieveAddressWithDB.Check RC':
('.RetrieveAddressWithDB_CheckRC.Main', '6.6') : Failed to navigate to path element number '3' because it does not exist.
----------------------------------------------------------------------------------------------
Note: There is a bug in our ESQL :-)
----------------------------------------------------------------------------------------------
2007-02-22 14:40:01.669821    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Check RC': Evaluating expression
''InputRoot.MRM.resns:raca.resns:rc'' at ('.RetrieveAddressWithDB_CheckRC.Main', '6.6'). This resolved to
''InputRoot.MRM.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:rc''. The result
was ''NULL''.
2007-02-22 14:40:01.669952    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Check RC': Evaluating expression
''InputRoot.MRM.resns:raca.resns:rc > 0'' at ('.RetrieveAddressWithDB_CheckRC.Main', '6.39'). This resolved to ''NULL >
0''. The result was ''NULL''.

----------------------------------------------------------------------------------------------
Still in the "Check RC" node here, this is the point where we set the name of the next label for the upcoming RouteToLabel
node:
----------------------------------------------------------------------------------------------

2007-02-22 14:40:01.670009    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''SET OutputLocalEnvironment.Destination.RouterList.DestinationData[1].labelname = 'RetrieveOK';'' at
('.RetrieveAddressWithDB_CheckRC.Main', '9.7').
2007-02-22 14:40:01.670075    10140   UserTrace   BIP2566I: Node 'RetrieveAddressWithDB.Check RC': Assigning value
'''RetrieveOK''' to field / variable ''OutputLocalEnvironment.Destination.RouterList.DestinationData[1].labelname''.
2007-02-22 14:40:01.670125    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Check RC': Executing statement
''RETURN TRUE;'' at ('.RetrieveAddressWithDB_CheckRC.Main', '12.3').
2007-02-22 14:40:01.670217    10140   UserTrace   BIP4007I: Message propagated to 'out' terminal of node
'RetrieveAddressWithDB.Check RC'.
```

```
---------------------------------------------------------------------------------------------
The RouteToLabel node behaves as expected:
---------------------------------------------------------------------------------------------

2007-02-22 14:40:01.670321    10140   UserTrace   BIP4241I: Message propagated to target Label node by RouteToLabel node
'RetrieveAddressWithDB.RouteToLabel'. A RouteToLabel node has received a message and is propagating it to the appropriate
Label node. No user action required.
2007-02-22 14:40:01.670349    10140   UserTrace   BIP4220I: Message propagated to out terminal from node
'RetrieveAddressWithDB.RetrieveOK'.

---------------------------------------------------------------------------------------------
The "RetrieveOK" label node takes control:
---------------------------------------------------------------------------------------------

A label node has received a message and is propagating it to any nodes connected to its out terminal. No user action
required.

---------------------------------------------------------------------------------------------
The last major processing step is to update the local CUSTOMER database with the new address details retrieve from the
RetrieveAddress CICS w-s call.
---------------------------------------------------------------------------------------------

2007-02-22 14:40:01.670423    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Update Customer': Executing
statement   ''DECLARE tns NAMESPACE 'http://schemas.xmlsoap.org/soap/envelope/';'' at ('.tns', '1.1').
2007-02-22 14:40:01.670472    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Update Customer': Executing
statement   ''DECLARE resns NAMESPACE 'http://www.ITSORA03.ITSORACA.Response.com';'' at ('.resns', '1.1').
2007-02-22 14:40:01.670519    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Update Customer': Executing
statement   ''DECLARE reqns NAMESPACE 'http://www.ITSORA03.ITSORACA.Request.com';'' at ('.reqns', '1.1').
2007-02-22 14:40:01.670564    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Update Customer': Executing
statement   ''BEGIN ... END;'' at ('.RetrieveAddressWithDB_UpdateCustomer.Main', '2.2').
2007-02-22 14:40:01.670608    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Update Customer': Executing
statement   ''UPDATE Database.ANDREWG.CUSTOMER SET ADDRESS1 = ..., ADDRESS2 = ..., SUBURB = ..., STATE = ..., POSTCODE =
..., HASH = ... WHERE ...'' at ('.RetrieveAddressWithDB_UpdateCustomer.Main', '3.3').
2007-02-22 14:40:01.671748    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressLine1'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '4.21'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:AddressLine1''. The result
was '''IBM Georgia'''.
2007-02-22 14:40:01.671880    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressLine2'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '5.19'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:AddressLine2''. The result
was '''1 Argonaut St '''.
2007-02-22 14:40:01.672003    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:Suburb'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '6.17'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:Suburb''. The result was
'''Tbilsi                                     '''.
2007-02-22 14:40:01.672127    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:State'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '7.16'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:State''. The result was
'''GEOR      '''.
2007-02-22 14:40:01.672245    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:Postcode'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '8.19'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:Postcode''. The result was
'''11111     '''.
2007-02-22 14:40:01.672371    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressHash'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '9.15'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
```

```
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:AddressHash''. The result was
''527817889''.
2007-02-22 14:40:01.672448    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:FirstName'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '10.26'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:FirstName''. The result was
'''Thomas'''.
2007-02-22 14:40:01.672526    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:MiddleName'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '11.25'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:MiddleName''. The result was
'''J'''.
2007-02-22 14:40:01.672598    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Update Customer': Evaluating
expression ''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:LastName'' at
('.RetrieveAddressWithDB_UpdateCustomer.Main', '12.23'). This resolved to
''Root.MRM.http://schemas.xmlsoap.org/soap/envelope/:Body.http://www.ITSORA03.ITSORACA.Response.com:ITSORA03OperationRespo
nse.http://www.ITSORA03.ITSORACA.Response.com:raca.http://www.ITSORA03.ITSORACA.Response.com:LastName''. The result was
'''Watson'''.
2007-02-22 14:40:01.672665    10140   UserTrace   BIP2544I: Node 'RetrieveAddressWithDB.Update Customer': Executing
database SQL statement ''UPDATE ANDREWG.CUSTOMER SET ADDRESS1 = ?, ADDRESS2 = ?, SUBURB = ?, STATE = ?, POSTCODE = ?, HASH
= ? WHERE (((FIRSTNAME)=(?))AND((MIDDLENAME)=(?)))AND((LASTNAME)=(?))'' derived from
('.RetrieveAddressWithDB_UpdateCustomer.Main', '3.3'); expressions
''Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressLine1,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressLine2,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:Suburb,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:State,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:Postcode,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:AddressHash,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:FirstName,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:MiddleName,
Root.MRM.tns:Body.resns:ITSORA03OperationResponse.resns:raca.resns:LastName'';resultingparametervalues'''IBMGeorgia
',  '1 Argonaut St                                                              ',  'Tbilsi
',  'GEOR     ',  '11111    ',  527817889,  'Thomas                              ',  'J
',  'Watson                                            '''.
2007-02-22 14:40:01.673701    10140   UserTrace   BIP2537I: Node 'RetrieveAddressWithDB.Update Customer': Executing
statement   ''RETURN TRUE;'' at ('.RetrieveAddressWithDB_UpdateCustomer.Main', '13.3').
2007-02-22 14:40:01.673792    10140   UserTrace   BIP4007I: Message propagated to 'out' terminal of node
'RetrieveAddressWithDB.Update Customer'.
2007-02-22 14:40:01.673874    10140   UserTrace   BIP2539I: Node 'RetrieveAddressWithDB.Log Success': Evaluating
expression ''Root'' at ('', '4.3'). This resolved to ''Root''. The result was ''ROW... Root Element Type=16777216
NameSpace='' Name='Root' Value=NULL''.


-----------------------------------------------------------------------------------------------
And finally, the 'Success' trace node writes a record to the local error log indicating all is OK. The following "Error"
for trace entry 19003 is side-effect of our final Trace node which writes to the local error log (Event Viewer) with Event
ID 19003. The trace formatter doesn't understand this message. This can be safely ignored.

From the local error log showing this (partial) entry:

The description for Event ID ( 19003 ) in Source ( WebSphere Broker v6002 ) could not be found. It contains the following
insertion string(s): .
AJGBROKER1.WSTest
'
Success !!

(
  (0x01000000):Properties        = (
    (0x03000000):MessageSet      = 'L849NPS002001'
    (0x03000000):MessageType     = 'Envelope'
    (0x03000000):MessageFormat   = 'XML1'
                :
-----------------------------------------------------------------------------------------------

2007-02-22 14:40:01.675413    10140   Error       ??????????   19003   ??????????   BIPv600.properties
```

```
2007-02-22 14:40:01.675639    10140   UserTrace   BIP4067I: Message propagated to output terminal for trace node
'RetrieveAddressWithDB.Log Success'.
                                       The trace node 'RetrieveAddressWithDB.Log Success' has received a message and is
propagating it to any nodes connected to its output terminal.
                                       No user action required.

End of Trace
```

A

Additional material

This section refers you to additional material that can be downloaded from the Internet.

Locating the Web material

The Web material associated with this book is available in soft copy on the Internet from the IBM Redbooks Web server. Point your Web browser to the following address:

```
ftp://www.redbooks.ibm.com/redbooks/SG247425
```

Alternatively, you can go to the IBM Redbooks Publications Web site at the following location:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247425.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
BrokerProject.zip	PIF Exports of the Broker sources

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

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 IBM Redbooks” on page 262. Note that some of the documents referenced here may be available in soft copy only.

- *Application Development for CICS Web Services*, SG24-7126
- *Securing Access to CICS Within an SOA*, SG24-5756
- *Patterns: Service-Oriented Architecture and Web Services*, SG24-6303
- *Implementing CICS Web Services*, SG24-7206
- *CICS Transaction Server V3R1 Channels and Containers Revealed, SG24-7227*

Other publications

These publications are also relevant as further information sources:

- *CICS Web Services Guide*, SC34-6458-04

Online resources

The following Web sites are also relevant as further information sources:

- XML Standards reference

  `http://www.w3.org/XML/`
- UDDI Standards

  `http://www.uddi.org/`
- WDz Web site

  `http://www.ibm.com/software/awdtools/devzseries/`

How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and additional materials, as well as order hard copy Redbooks or CD-ROMs, at the following Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

Developing Web Services Using CICS, WMQ, and WMB

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Developing Web Services Using CICS, WMQ, and WMB

Bottom-up application design and re-use of traditional code

Exposing applications as Web services

Modern tooling techniques

This IBM Redbooks publication provides a practical demonstration of how to develop applications that take advantage of CICS Web services facilities. This book can be viewed as a follow-on from the IBM Redbooks publication *Application Development for CICS Web Services*, SG24-7126-00, with the addition of using modern tooling techniques. Because we are creating a new application, we follow the bottom-up approach described in *Application Development for CICS Web Services*, SG24-7126-00. Although not a requirement, we highly recommend that you review that publication for a much deeper discussion of CICS Web services development topics and alternative approaches.

The primary purpose of this book is to demonstrate that well structured CICS Web services are easy to develop using the CICS Web Services Assistant. We also look at modern tooling, such as WebSphere Developer for zSeries (WD/z).

SG24-7425-00 ISBN 0738489239