CICS Transaction Server
White paper

WebSphere software

# Service Flow Feature of CICS Transaction Server for z/OS, Version 3.2

*Overview and business usage scenarios*

*By Nick Carlin, IBM Software Group, nick_carlin@uk.ibm.com*
*and William Alexander, IBM Software Group, walexand@us.ibm.com*

January 2008

**Contents**



## Introduction

Many leading enterprises use IBM System z™ mainframe servers running IBM CICS® Transaction Server to provide industrial strength, mission-critical applications for their businesses. The proven strengths of CICS running on IBM mainframes have, over time, led to a huge investment in CICS application code and skills. However, to be agile in today's fast-moving marketplace, organizations must embrace new technologies while leveraging these existing assets.

This white paper is aimed at technical professionals with an interest in Service Flow Feature, integrated in IBM CICS Transaction Server for z/OS®, Version 3.2. The paper provides an overview of the Service Flow Feature and its latest enhancements, and provides five scenarios that show how clients have used the feature in their environments.

## Service Flow Feature: A strategic solution for application reuse and modernization

Service Flow Feature helps to reduce cost and risk, while speeding application and service delivery by enabling you to reuse assets rather than rebuild them.

Service Flow Feature achieves this by exposing existing applications as services to be used across departments and organizations to enable business process automation in alignment with a service oriented architecture (SOA).

Service Flow Feature was first introduced with CICS Transaction Server for z/OS, Version 3.1, comprising the Service Flow Modeler tooling component and Service Flow Runtime component. Version 3.2 of CICS Transaction Server for z/OS continues to offer the feature with additional enhancements.

Together, the tooling and runtime components of the Service Flow Feature support application reuse and modernization. They enable applications and services external to CICS to use and integrate existing CICS 3270, CICS communication area (COMMAREA), CICS channels and containers, and other applications seamlessly as callable services (see Figure 1).

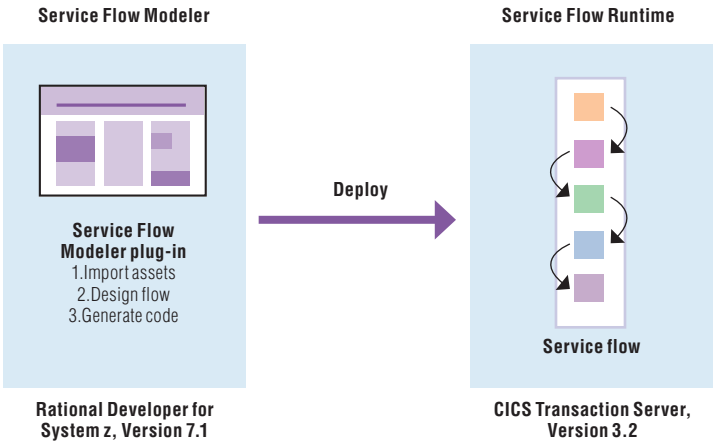


*Figure 1. CICS Transaction Server, Version 3.2 Service Flow Feature components*

Before Service Flow Modeler became available, there were tools that could rapidly convert single CICS applications to Web services. But no tools were available that would allow you to aggregate multiple applications together into a more-suitable business service. In addition, there was no easy approach to creating a business service that had a mixture of terminal and COMMAREA applications.

Service Flow Feature in CICS Transaction Server can work with existing CICS applications that do not adhere to best-practice programming patterns and have presentation routines, business data and business logic embedded together. Many terminal applications running today have these characteristics. Such applications are expensive to update and maintain, so reusing them as a whole in a broader business transaction is more cost-effective and less error-prone than rewriting them.

Note: IBM WebSphere® Host Access Transformation Services (HATS) is a product from IBM that also works with terminal applications. Service Flow Feature and HATS are compared and contrasted later in this document.

Service Flow Feature has the ability to integrate multiple applications into a single service. This activity, also known as *aggregation*, has advantages over explicitly chaining services together. Aggregation reduces complexity by freeing the service invoker of the concern of managing callouts to individual applications in the chain. Ultimately, aggregation at the runtime level reduces the execution cost of the transaction through reduced network overhead, reuse of existing assets and application simplification.

The technique used with Service Flow Feature is *noninvasive*, meaning that CICS application assets accessed by the service flow do not have to be altered in support of the service flow (see Figure 2).

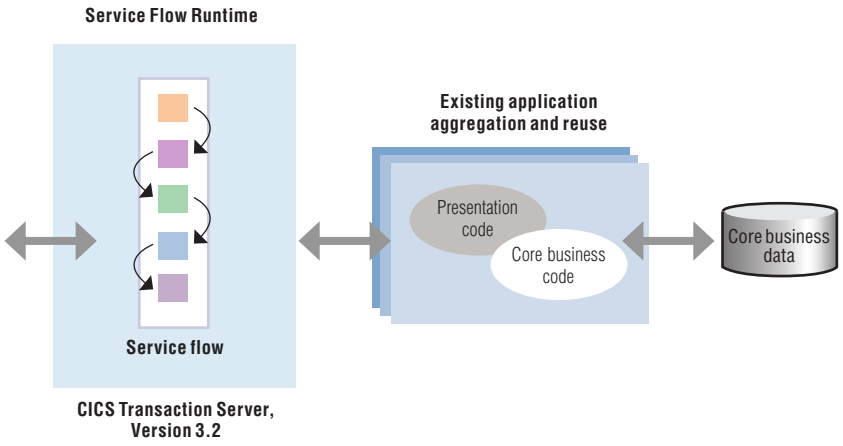


*Figure 2. Reuse and aggregation using Service Flow Feature*

Service Flow Feature is an optional, no-charge feature of CICS Transaction Server for z/OS, Version 3.2. IBM Rational® Developer for System z, Version 7.1 delivers the required Service Flow Modeler tooling support. CICS Transaction Server for z/OS, Version 3.1 also continues to offer Service Flow Feature.

*Service flows*

A service flow is a short-running, noninterruptible microworkflow without the ability to be suspended or resumed.

The following example shows how a service flow can control the execution of existing applications in an enterprise.

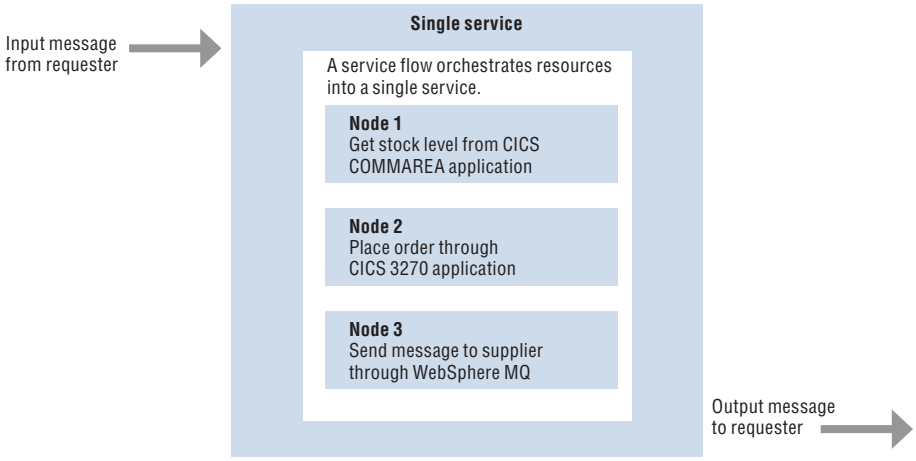


*Figure 3. Example service flow*

Figure 3 is a graphical representation of the sequence of activities performed to implement a business service of an enterprise. In the Service Flow Modeler tooling, a service flow comprises a graph of nodes with defined entry and exit points. Each node represents invoking a service operation, controlling the sequence of operations or performing reusable business logic. The service flow can be altered depending on the output of any of the nodes. For example, if the customer number used in Node 2 of Figure 3 is not valid, the service flow can be modeled to return a message to the requester indicating that the customer number was invalid and that a new one must be provided in a subsequent request.

Invoking a CICS service flow

A CICS service flow is invoked by a *service requester* – the application that looks for and initiates an interaction with a service. Examples of service requesters are Web services, applications enabled by IBM WebSphere Process Server and IBM WebSphere MQ, such as IBM WebSphere Message Broker.

Each service requester uses an interface to the common point of entry for a CICS Service Flow Runtime, the DFHMADPL program (see Figure 4).

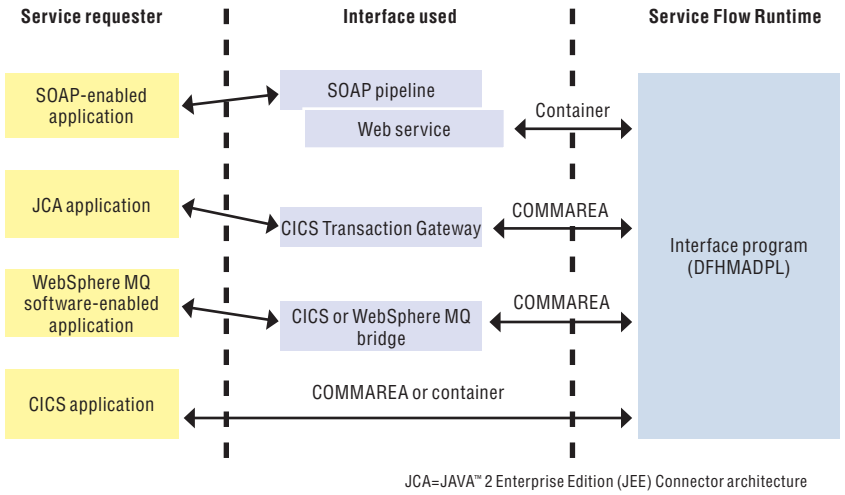


*Figure 4. Invoking a CICS service flow*

A common method of invoking a service flow is from a Web service using CICS Web services support. In this case, the application handler in the CICS pipeline passes the appropriate container to the CICS Service Flow Runtime interface program, DFHMADPL.

Service flows can also be invoked through CICS-supplied interfaces, such as the External Call Interface (ECI), the External CICS Interface (EXCI) and through Distributed Program Link (DPL). These methods could potentially be used with CICS Transaction Gateway. When using a CICS-supplied interface, the request can only be processed in synchronous mode. Synchronous requests require the service requester to wait until the service flow has completed processing before it can continue with further activities.

WebSphere MQ software-enabled service requesters might also invoke a flow. In addition to synchronous mode processing, asynchronous mode processing is supported when using a WebSphere MQ software-enabled service requester. Asynchronous requests allow the service requester to continue with further activities while the service flow is processing, and then get the response when it is delivered upon service flow completion.

*Service Flow Runtime*

The service flow interface program, DFHMADPL, creates a CICS Business Transaction Services (BTS) process to run the service flow. So, CICS BTS must be enabled in the CICS region for Service Flow Runtime to use the CICS Service Flow Feature.

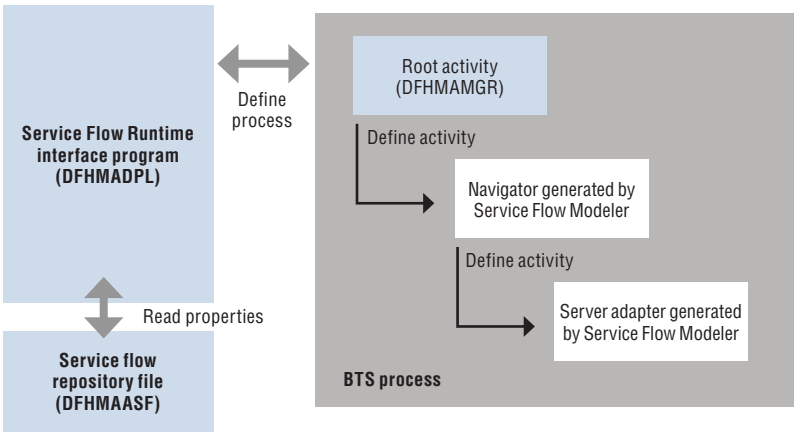


*Figure 5. Service flow running in a BTS process*

In Figure 5, a service flow is invoked when a service requester sends a request message to the CICS Service Flow Runtime. The CICS Service Flow Runtime interface program, DFHMADPL, receives the request message and reads the service flow repository file, DFHMAASF. DFHMAASF contains a record for every service flow that is deployed. DFHMADPL puts data from the original request and the information that is defined for the service flow into data containers of CICS BTS and starts a process in CICS BTS with the navigation manager, DFHMAMGR, as the root activity.

If the service flow is simple and contains one server adapter, the navigation manager runs the server adapter directly as a child activity. If the service flow is more complex, the navigation manager initiates a flow navigator as the child activity.

### Service flow adapters

Service flow adapters enable the service flow to call other business applications within the enterprise. A service flow can coordinate and orchestrate the use of other programs in the enterprise.

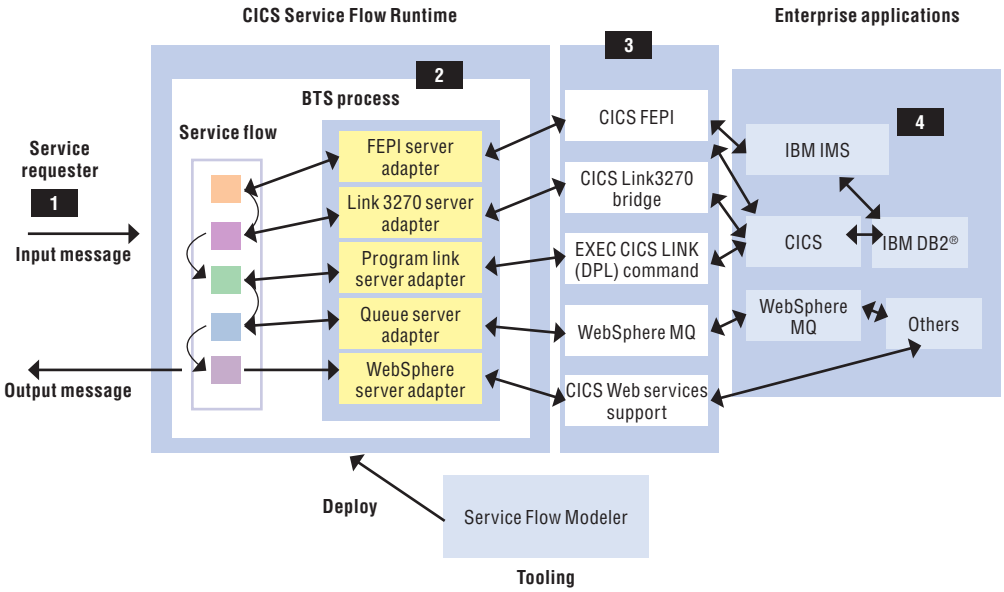


*Figure 6. Service flow adapters*

Figure 6 illustrates the following sequence of events for use of service flow adapters:

1. *An input message arrives from the service requester.*
2. *CICS Service Flow Runtime initiates a flow navigator as the child activity, as described in the previous section about Service Flow Runtime. The flow navigator runs server adapters in the correct order, as determined by the service flow. The following optional server adapters can be defined:*

*a. FEPI server adapter*

A front-end programming interface (FEPI) server adapter performs screen navigation using FEPI support in CICS. Using FEPI commands, it sends requests to and receives replies from any CICS (or potentially IBM IMS™) application with a 3270 data stream. So, using FEPI, it is possible to drive an IMS transaction, for example, using the CICS Service Flow Feature.

*b. Link3270 server adapter*

The Link3270 server adapter performs screen navigation using the CICS Link3270 bridge mechanism. The Service Flow Modeler contains the 3270 emulation and navigation logic for 3270 application screens that use basic mapping support (BMS) maps or 3270 data streams.

*c. Program link server adapter*

The program link server adapter performs programming links to CICS applications using the EXEC CICS LINK command.

*d. Queue server adapter*

The queue server adapter handles the requests and responses for WebSphere MQ software-enabled applications.

*e. Web service server adapter*

The Web service server adapter performs outbound Web service requests using the existing Web services support in CICS.

3. *The mechanisms for accessing existing business applications will be either by FEPI, Link3270 Bridge, EXEC CICS LINK, WebSphere MQ or EXEC CICS INVOKE WEBSERVICE.*
4. *Existing business applications could reside on the same system as the CICS Service Flow Runtime or on different systems in the enterprise accessible over a network.*

Note: Although CICS Service Flow Runtime has a prerequisite to run in CICS Transaction Server, Version 3.1 or 3.2, there is no such restriction for any target CICS application regions where user transactions and applications are running.

Key point: When a service request comes to CICS, CICS Service Flow Runtime navigates the appropriate 3270 screen sequences, formulates a consolidated response, and sends a single service response to the requester. Flow sequences for Link3270 Bridge, FEPI, DPL (COMMAREA and channels and containers), and WebSphere MQ might be aggregated in a single service request.

*Service Flow Modeler*

Service Flow Modeler is a component of the Enterprise Service Tools within Rational Developer for System z (see Figure 7) supporting the transformation and reuse of existing applications. Service Flow Modeler enables the move towards an SOA.


*Figure 7. Service Flow Modeler project in Rational Application Developer for System z*

Service Flow Modeler is a complementary approach to CICS Web services. The assumption with Web services is that the service we are exposing in CICS has a clean and precise COMMAREA interface and has been designed and programmed so that the presentation and business logic processing are totally separated. Without any additional application programming, Service Flow Modeler will generate any required program code to enable extraction and aggregation of existing applications, even if the business logic exists in the presentation layer.

The Service Flow Modeler toolset is layered and is based on a common information model. The information model is based largely on MXSD (which is a variant of XML Schema Definition [XSD] files) and Web Service Definition Language (WSDL) documents to describe operations together with their input and output.

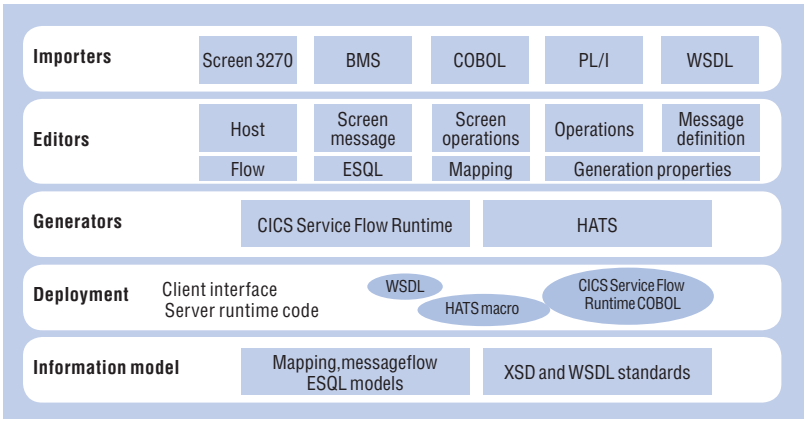


*Figure 8. Service Flow Modeler components*

As we have seen, the service flow comprises a number of activity nodes orchestrated by Service Flow Runtime. The following nodes illustrate the different types that can be modeled with Service Flow Modeler:

- *Screen-based nodes*
- *Links to other programs in CICS (through a DPL)*
- *Request and response to WebSphere MQ software-enabled programs*
- *Outbound Web service requests*

The service flow is built as a series of connected nodes in an Eclipse-based graphical workspace running in Rational Developer for System z.

Figure 9 shows an input order to the CICS catalog sample application supplied with CICS. Interactions with two 3270 screens (select from menu and order item) have been recorded using the Service Flow Modeler record function.

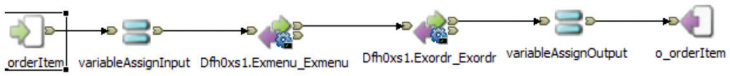


*Figure 9. Example flow in Service Flow Modeler*

Figure 10 summarizes the various operations that can be made on nodes.

| Icon | Operation | Definition |
|---|---|---|
| | Receive | Input node for flow |
| | Reply | Output node from flow |
| | Throw | Throw fault node |
| | Invoke | Generic invoke (target not yet defined |
| | Invoke operation | Invoke a non-terminal |
| | Invoke terminal | Invoke a terminal screen operation |
| | Invoke web service | Invoke a web service request |
| | Invoke flow | Invoke a flow |
| | Assign | Map data between messages in the flow |
| | Switch | Flow-control decision node |
| | While | Flow-control loop node |

*Figure 10. Service Flow Modeler icon descriptions for node operations*

### CICS Service Flow Feature usage scenarios

There are a number of ways that Service Flow Feature can be used in CICS Transaction Server for z/OS. The following scenarios are based on discussions with clients about how they plan to use, or are using, Service Flow Feature in their environments.

*Scenario 1: Retail*

The problem

The client is a large catalog retailer who supplies information to an external third party. The third party calls a CICS COMMAREA application within the retailer, which returns one item of data only. All was well until the third party changed its requirements and needed 10 items of different information from the retailer. This requirement was implemented and resulted in 10 calls to the retail system instead of one, resulting in an impact to CICS performance and significantly increased network overhead. One of the emerging problems with SOA implementation is *chatty services* – it is possible to expose *too much*, which can lead to the significant overhead of driving many separate services.

The solution

To implement an optimal method of integration between modern enterprise solutions and an existing enterprise system, the number of interactions between the systems should be minimized. This reduces the volume of request data that is being exchanged over the transport, and it reduces the cost of data content that is processing in each system. To minimize interactions, the client implemented a *coarse-grained* interaction pattern across the external system boundary complemented by highly optimized processing of the request payload within the bounds of each system. As a result, the client changed the granularity of the 10 calls into one call (see Figure 11). The client also used Service Flow Feature to manage the collection of information and pass it back to the requester in a single invocation of a service flow.

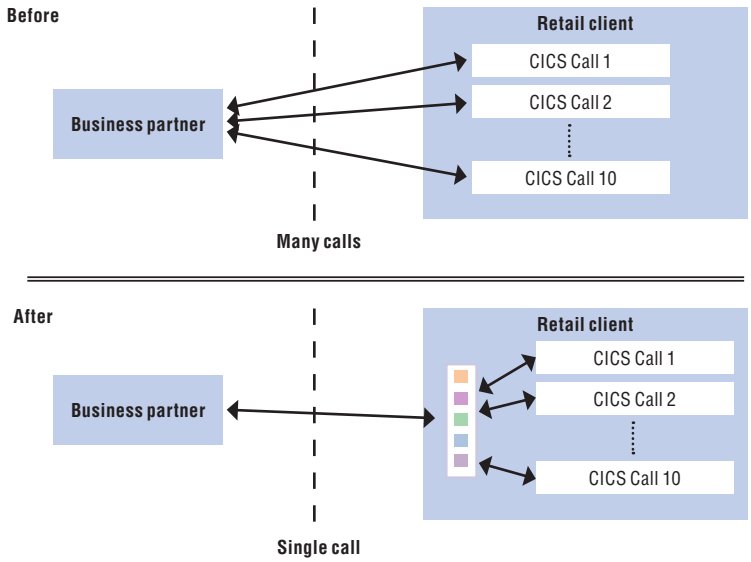


*Figure 11. Scenario 1: Aggregation to single call*

This solution illustrates a key element of the broader SOA picture; services exposed in a way that aligns with the business perspective. In this case, the client's business partner needed 10 items, and initially it was implemented as simply 10 separate calls. It works, but in reality the client really wanted to view it as a single call, with 10 item numbers inside. Service Flow Feature in CICS Transaction Server helps achieve the business goal; one request, one answer, with that one answer containing information about 10 items.

### *Scenario 2: Financial services*

#### The problem

A banking client wanted to reuse an existing 3270 terminal-based CICS application in support of a new business requirement. However, the application required well-formed user data. Until now, this user data was validated by the presentation-layer logic of the application. The client did not want to reuse the same presentation interface but could not access the back-end application without it.

#### The solution

The client used Service Flow Feature to reface the application for Web services access. Service Flow Modeler enables reuse of application code that was intended for a 3270 environment in which the presentation and business logic layers are irretrievably intertwined. Even better, it enables the client to selectively extricate particular screen fields from more than one screen and pass the resulting data aggregation back to the calling application interface (see Figure 12).

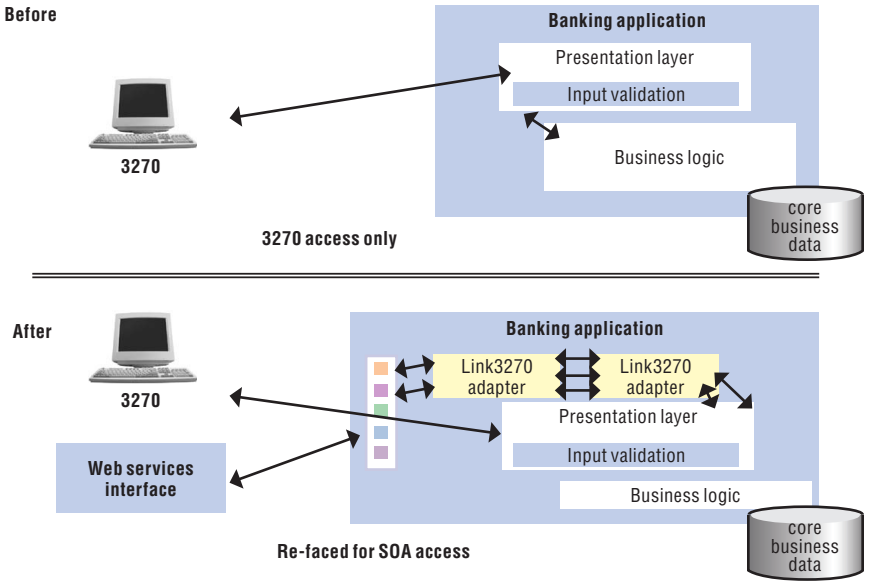


*Figure 12. Scenario 2: 3270 application re-faced for SOA access*

The client could now use a modern interface that passed information from users to the presentation logic of the CICS application. The presentation logic then carried out the correct validation of user input before calling the existing core business logic. The application, although it did not conform to modern programming standards, did not need to be modified, thus avoiding development effort and cost. During migration, the existing 3270 interface was used unchanged. In addition, risk was lowered and speed of delivery was optimized due to the reuse of proven input validation logic.

### *Scenario 3: Financial services*

The problem

A banking client had an audit application that logged requests from multiple, different applications. The logging request could come from CICS (COMMAREA-based) or WebSphere MQ software-based applications and would be logged to separate data stores. The client wanted to join these logging requests to a single log and was considering writing COBOL code to join the logging processes. But it was considered too expensive to write and maintain COBOL code for this purpose.

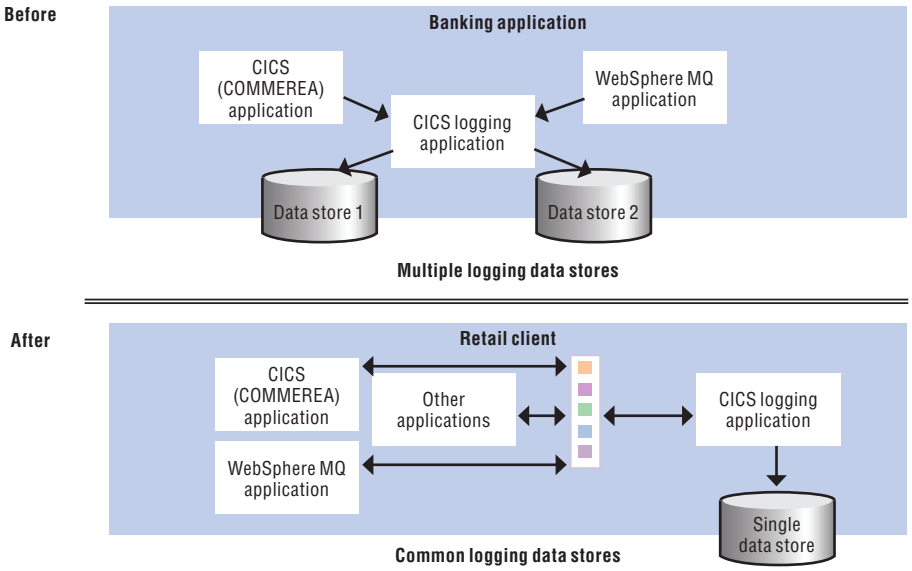


*Figure 13. Scenario 3: Common logging*

The solution

The client used Service Flow Feature to generate a service flow to merge the logging requests. This avoids writing and maintaining COBOL code. Now a single unified log is written with messages from applications in timestamp order, which aids with problem determination (see Figure 13). The possibility now exists to add new applications to this logging infrastructure with minimum effort.

*Scenario 4: University*

The problem

The client wanted to adopt an SOA to improve the speed and responsiveness of the IT organization to meet changing business requirements. A major issue was that vital student information was stored in multiple places, and copies of this information could become desynchronized, which could cause potential embarrassment to the university and possible loss of revenue.

The solution

The university used Service Flow Feature to generate a service that can retrieve and update the student data records respectively. The service can also take actions to update copies of vital information wherever they reside in the various satellite systems.

In addition to using Service Flow Feature, the client also chose to implement WebSphere Message Broker as an enterprise service bus (ESB) for transforming the disparate message formats between the requesting applications and the new service. Eventually, when the effort is completed, the remote copies of the data will be retired, and the university will have a single system of record for student data without changes to the calling application queries (see Figure 14).

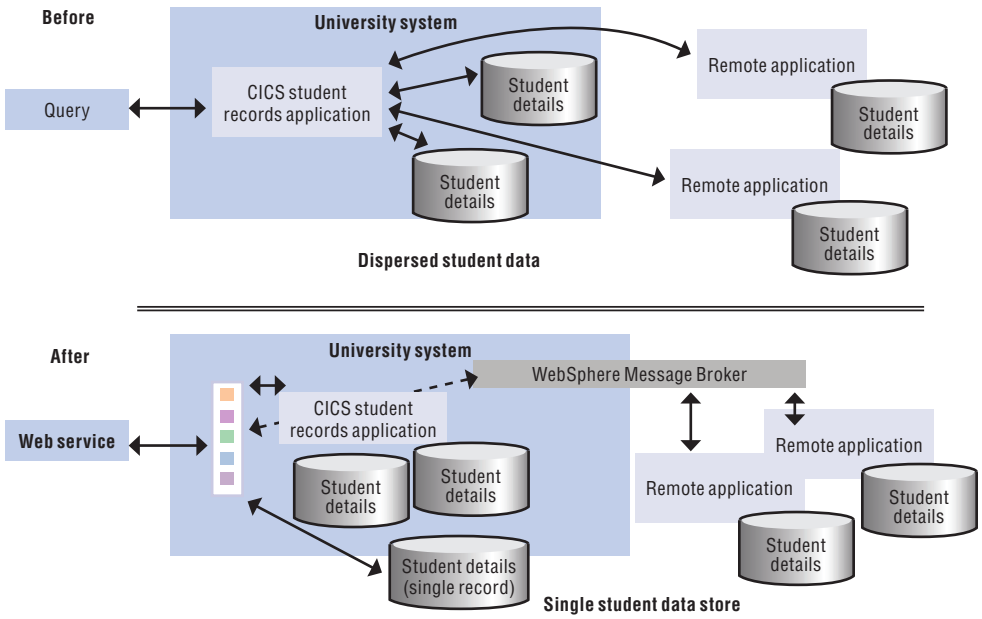


*Figure 14. Scenario 4: Single data store using Service Flow Feature and WebSphere Message Broker*

### *Scenario 5: Transportation*

The problem

The client was experiencing a high operator-turnover rate, high training costs and poor operator performance due to a 3270 screen-based application that was antiquated, but core to the business. This application displayed much of the data in codes in attempts to pack the limited, screen real estate with useful information. In addition, related information was dispersed throughout the application because of previously overcrowded screens. To gather necessary information, users needed to navigate through multiple screens.

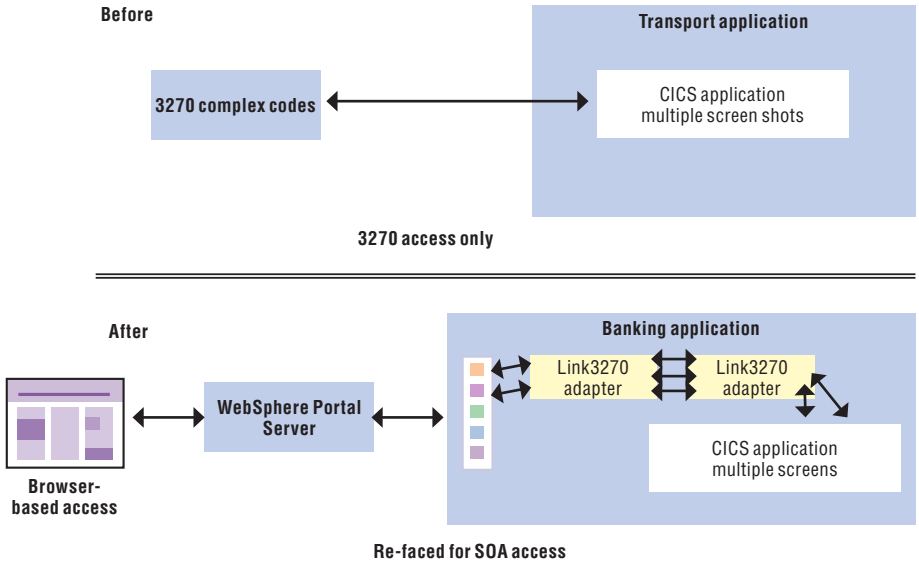


*Figure 15. Scenario 5: Complexity reduction with Service Flow Feature and IBM WebSphere Portal*

## The solution

The transportation client used Service Flow Feature to consolidate the numerous screens and data items into callable services, based on business function. These services are invoked by a new Web portal, reducing training, turnover and mistakes for new personnel, because the client no longer needs to perform complex navigation through multiple screens to get the data and analyze it.

### *Service Flow Feature and HATS*

HATS from IBM can support transforming 3720- and 5250-terminal data streams and BMS maps to HTML. HATS enables a user to record existing terminal-based applications and transform them to JavaServer Pages (JSP) or Enterprise JavaBeans (EJBs) directly. HATS requires the HATS toolkit plug-in to Rational Application Developer for tooling and WebSphere Application Server to host the HATS run time. HATS can also be invoked from a Web service.

In addition to terminal-based modernization provided by HATS, the Service Flow Feature can handle COMMAREA, channels and containers, and direct interfaces to WebSphere MQ applications. Unlike HATS, for 3270 terminal applications the Service Flow Feature uses the Link3270 adapter and bridge. The Link3270 mechanism does not ship representations of the entire screen. It only sends the modeled fields and so is potentially more efficient.

An existing application can be modeled with Service Flow Modeler and then deployed to HATS for testing. The flow of the application could be debugged and the expected behavior verified. If reuse of this application is needed, it could be deployed directly to the Service Flow Runtime of CICS Transaction Server for z/OS to enable better efficiency, performance and lower overhead than when used in the HATS run time.

HATS can run Web services created with any supported level of CICS Transaction Server. The HATS approach might benefit CICS Transaction Service customers who have not yet migrated to a release of CICS that supports the Service Flow Runtime (CICS Transaction Server, Version 3.1 or 3.2) or who have limited availability of CICS skills. The assets recorded with Service Flow Modeler can, at a later date, be redeployed directly into CICS Transaction Server, Version 3.2 Service Flow Runtime, enabling easy, forward migration.

The other possible advantage of CICS Transaction Server for z/OS Service Flow Feature over HATS is that Service Flow Feature does not require an IBM WebSphere Application Server infrastructure. For clients who have such an infrastructure, adding HATS is relatively straightforward. For clients with just CICS, the Service Flow Feature can be easier if the objective is to perform application aggregation with a Web services interface.

### Service Flow Feature enhancements in CICS Transaction Server for z/OS, Version 3.2

In CICS Transaction Server for z/OS, Version 3.2, Service Flow Runtime delivers a set of capabilities that support the enhanced Service Flow Modeler to increase business flexibility.

Rational Developer for System z, Version 7.1 is required for building flows to run in the CICS Transaction Server for z/OS, Version 3.2 Service Flow Runtime, and it can also be used to deploy service flows into CICS Transaction Server for z/OS, Version 3.1 Service Flow Runtime. Without making any changes, existing flows in Version 3.1 can be regenerated and redeployed to the Version 3.2 runtime environment.

The process of deploying service flows now includes the flexibility to define resources dynamically using a new service flow repository file (DFHMAASF). In addition you can now manage the deployed service flows using a management transaction (CMAN). This provides the new capability to discard a service flow when it is no longer required or to disable a service flow temporarily, restricting access to particular business services until a preferred time.

An additional processing mode, called *Link mode*, has been added to Service Flow Runtime. Link mode enables a synchronous service flow to run in a single task and unit of work. This improves performance by using fewer tasks during request processing. A failure in any server adapter results in the unit of work being rolled back, including all work performed by other server adapters in the service flow. This mode provides greater flexibility when deciding how to run service flows in CICS.

*Enhancements to service adapters*

The server adapters that are supported in this release are enhanced to provide support for additional functions.

- FEPI server adapter. *Generated FEPI server adapters are now smaller in size and support screen recognition, as modeled in Service Flow Modeler. In addition, the restriction on the screen sizes that the FEPI adapter supports has been lifted.*
- Link3270 server adapter. *The Link3270 server adapter now supports target applications that use a 3270 data stream to send screen buffers, as well as BMS maps and text during service flow processing. The Link3270 server adapter now supports the following CICS API commands: SEND, RECEIVE and CONVERSE. Vector logging also now supports 3270 data streams.*
- Program link adapter. *DFHMASDP, referred to in previous releases as the DPL server adapter, has been renamed and is now referred to as the program link adapter. DFHMASDP now supports channels and containers as a method for passing data to a target application, in addition to the existing support for COMMAREAs. DFHMASDP now creates a channel if this interface is modeled in the service flow, populates the containers from BTS data containers, and passes them on the channel to the target application using an EXEC CICS LINK command. The program link adapter uses new BTS data containers to populate the containers on the channel.*
- Queue server adapter. *The queue server adapter DFHMASCQ performs all requests to WebSphere MQ software-enabled applications that are modeled in a service flow. This server adapter replaces the generated WebSphere MQ adapters from the previous release, reducing the number of generated programs that require deployment from Service Flow Modeler.*
- Web service server adapter. *The default transaction (CMAO), under which the Web service server adapter runs, can now be overridden in Service Flow Modeler.*

**Summary**

To run the Service Flow Feature on CICS Transaction Server for z/OS, Version 3.2, the following software is required:

- *Service Flow Feature of CICS Transaction Server, Version 3.2*
- *IBM Rational Developer for System z, Version 7.1*

Service Flow Feature along with its components, Service Flow Runtime and Service Flow Modeler, provides the following benefits:

- *Enables reusing and transforming existing applications, which is more cost-effective than writing new applications. Reusing proven applications can lower the risk of implementation and speed delivery time. In addition, extending existing applications where the source code is no longer available or can no longer be maintained can now be achieved in a cost-effective manner.*
- *Enables simplification and ease of use of existing business applications through aggregation of many transaction calls into a single call. Fine-grained applications become coarse-grained services through a process of aggregation, which can lower costs while maintaining qualities of service. Service Flow Feature can aggregate both 3270-based applications and those based on COMMAREA, or channels and containers.*

- *Unlocks existing applications, enabling them to join an SOA to speed business process integration and enable better responsiveness to changing business environments and emerging business opportunities. This also has the benefit of fostering SOA skills in traditional developers.*
- *Provides an easy-to-use graphical tool for creating service flows. By using the tool, a library of reusable, annotated components representing current assets can be built. The tooling generates the necessary integration code, which avoids the need for clients to write their own.*

New functions provided with CICS Transaction Server, Version 3.2 Service Flow Feature help ensure continued management and performance of a reuse strategy, which enables lowering cost and risk and speeding time to delivery of new applications in an enterprise.



IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Printed in the U.S.A.
01-08


WSW14012-USEN-00