CICS Transaction Server for z/OS
5.5

Using CICS Service Flow Runtime

Note

Before using this information and the product it supports, read the information in "Notices" on page 187.

This edition applies to the IBM® CICS® Transaction Server for z/OS® Version 5 Release 5 (product number 5655-Y04) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this PDF

CICS Service Flow Runtime supports service flows that are modeled, generated, and deployed using the service flow project tools plug-in of the IBM Developer for Z tool. This PDF describes how to set up, administer, and manage CICS Service Flow Runtime.

For details of the terms and notation used, see Conventions and terminology used in the CICS documentation in IBM Knowledge Center.

Date of this PDF

This PDF was created on 2024-04-22 (Year-Month-Date).

Chapter 1. Product overview

CICS Service Flow Runtime (CICS SFR) is the server runtime environment to the service flows that are modeled, generated, and deployed using the service flow project tools in the IBM Developer for Z product.

A *service flow* is a reusable composed business function that exposes a programming interface to a service requester in an Enterprise Information System (EIS). Depending on how the service flow is modeled, it can contain a wide variety of function, such as:

- Sequential navigation
- Conditional branching including decision and iteration
- Data typing
- Storing data context
- Transformation of data elements
- Logical operations
- Custom code.

Service flows are implemented in two stages:

At build time
A developer uses service flow project tools for the following tasks:

1. Model a newly composed business function as a service flow, using processes or services and their interfaces.
2. Capture existing EIS interfaces, such as screens or communication areas.
3. Generate the runtime code that is required to deploy the service flow into CICS Service Flow Runtime.
4. Deploy the service flow into CICS.

At run time
A service requester invokes the deployed service flow to perform a business function as modeled in the service flow project tools. By composing EIS interfaces and exposing the resulting service flows, it enables you to transform or adapt the enterprise to a new set of operations and methods that move applications towards a service-oriented architecture (SOA).

The deployed service flow adheres to a deployment pattern that describes whether it is simple or complex, depending on the number and type of server adapters that are required to run the service flow. A *server adapter* is a program that performs a particular function that was modeled in the service flow; for example, a Web service request or a link to a target application program. There are two types of service flow:

- A simple service flow contains only one server adapter. It normally represents a business function that involves simple screen sequencing or uses a distributed programming link to access a target application.
- A complex service flow can contain many server adapters that interact with different target applications to perform different processing, such as updating data.

CICS Service Flow Runtime can support the processing of service flows by using CICS Business Transaction Services (BTS).

Note: Structured fields, with the exception of QUERY PARTITION are not supported.

The runtime environment and BTS

CICS Service Flow Runtime uses the services provided by BTS to run service flows as business transactions in CICS.

An instance of a running business transaction in CICS is called a *process*. A process is a collection of BTS activities, where each activity is mapped to a traditional CICS transaction. The overall progress of the business transaction is controlled through a top-level program in the process that is called the *root activity*. The root activity in CICS Service Flow Runtime is the Navigation Manager (DFHMAMGR).

A service flow is invoked when a service requester sends a request message to the CICS Service Flow Runtime. The CICS SFR interface program DFHMADPL receives the request message and starts the Navigation Manager as the root activity, passing it the data from the request message. The Navigation Manager uses this data to run the required service flow as a hierarchical set of child activities, where data is exchanged between the activities in named areas of storage called *data-containers*. 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 complex, the Navigation Manager initiates a flow navigator as the child activity. The flow navigator ensures that the server adapters are run in the correct order as determined by the service flow, managing each adapter as a child activity.

The following figure shows the components that are used when a service requester invokes a complex service flow.

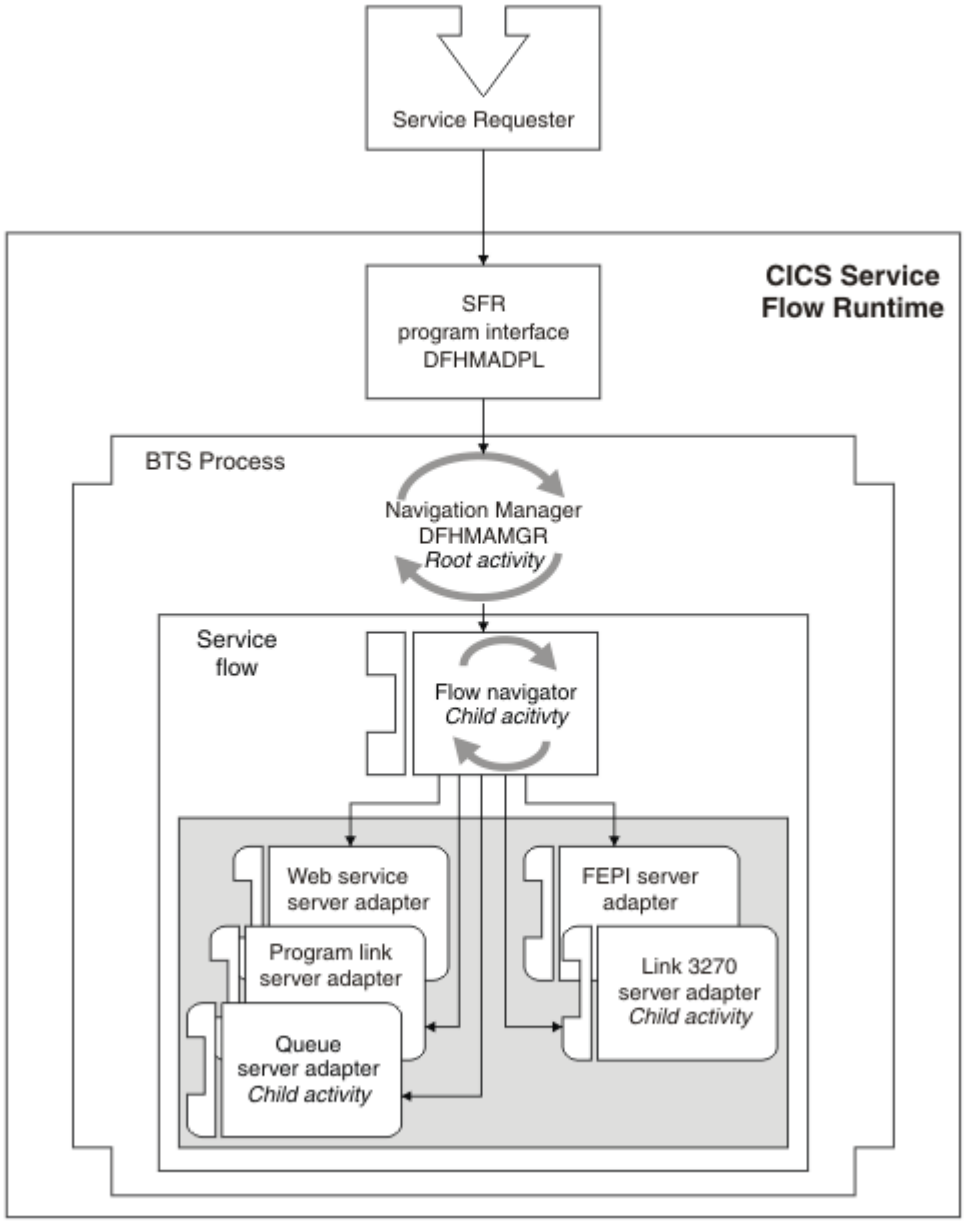


Figure 1. The components used in request processing for a complex service flow

CICS uses the BTS repository to store information about the activities and processes that are running in CICS Service Flow Runtime. You must have one BTS repository for each instance of CICS SFR. You cannot share the repository between CICS regions.

The CICS Service Flow Runtime enables any application that can initiate a CICS program to access these applications:

- CICS applications in many regions using a distributed program link (DPL)
- CICS and IMS applications using a 3270 data stream
- WebSphere® MQ-enabled applications using WebSphere MQ
- Web service providers and requesters

Several examples of service flows that implement business transactions are:

- Adding a sales order
- Checking an account balance

- Updating a customer record

The service flow project tools

The service flow project tools are part of the IBM Developer for Z product. You can use the tooling to create new services from existing applications in which the enterprise has invested substantially.

The service flow project tools consist of several major components that are shown in the following figure.

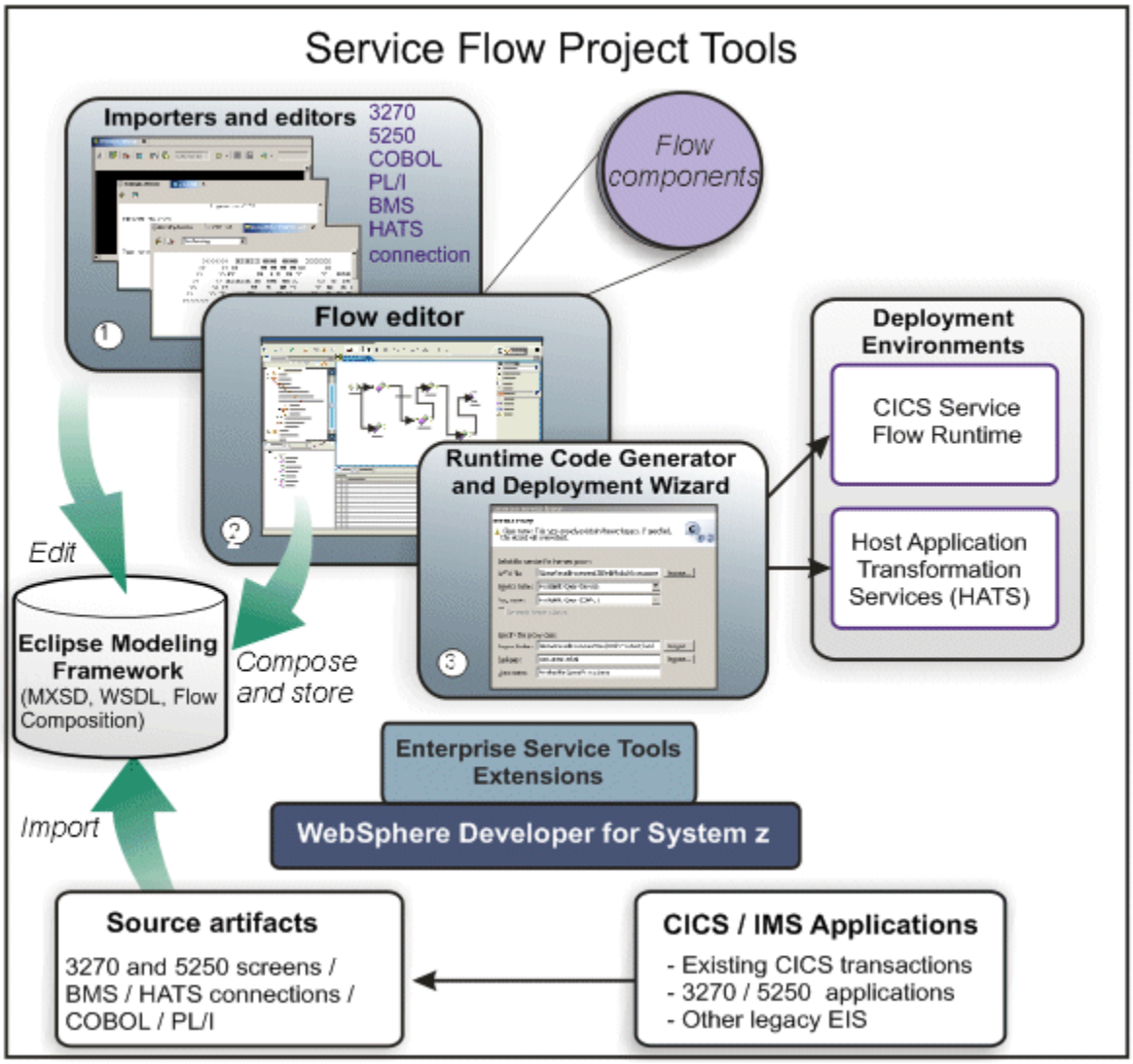


Figure 2. Components of Service Flow Modeler

Importers and editors

The importers enable you to import application resources from an existing Enterprise Information System (EIS) to represent these resources in a common information model. You can import the following resources:

- 3270 screens from CICS 3270 applications
- 5250 screens from 5250 applications on OS/400® systems
- COBOL record descriptions from existing CICS transactions
- BMS source code to build application data structures (ADS)

The editors enable you to control what is imported, as well as modifying the imported resources, modeling service flows and thus saving work.

The Flow editor
The Flow editor allows you to manually construct a service flow that represents a dialog and populate a service flow using a captured WSDL dialogue. Alternatively, you can use the Flow editor to annotate a service flow with alternative error paths and additional behaviors that cannot be captured using the importers.

Runtime code generator and deployment wizard
When you have finished modeling your flow, you can use the wizard to generate the runtime code that is required to deploy the service flow into CICS Service Flow Runtime.

For more information about the service flow project tools, see the relevant sections of the help in IBM Developer for Z.

Benefits of CICS Service Flow Runtime and service flow project tools

CICS Service Flow Runtime and service flow project tools provide the following benefits:

- The service flow project tools and the CICS Service Flow Runtime enable an organization to unlock critical IT assets (applications and data) from their current legacy status, and reworks those assets to participate in a service oriented architecture (SOA). This process is sometimes referred to as *application transformation*.

  When an organization has to create a new process for doing business with partners, suppliers, customers, or employees, service flow project tools and CICS Service Flow Runtime provide tools and supported runtime modules that allow the organization to take advantage of the value they already have in their existing enterprise information systems and to use these systems for service oriented business processes. Transforming existing applications is more efficient than creating new applications.
- The service flow project tools consists of a toolset that enables developers to analyze existing 3270 applications to understand where the capabilities lie within existing assets to reuse applications in new processes, services, or offerings.
- service flow project tools makes use of the Eclipse Integrated Development Environment (IDE).
- The service flow project tools generates all modeled code, runtime properties, and compilation JCL.
- The service flow project tools and the CICS Service Flow Runtime provide an efficient service flow integration in the z/OS environment.
- The CICS Service Flow Runtime allows service flows to be placed closer to the CICS and IMS EIS target applications, enabling a service requester to access multiple transactions or applications with one request.
- The CICS Service Flow Runtime enables more efficient use of computing resources by offloading work from the service requester. At run time, instead of a service requester invoking each transaction individually, it can send a request that performs the following processing:
  - Invoke CICS and IMS transactions, CICS applications, or WebSphere MQ-enabled applications.
  - Invoke Web service providers and requesters.
  - Handle all of the request processing.
- CICS Service Flow Runtime uses CICS business transaction services (BTS). CICS BTS makes it easier to model, control, and execute complex business transactions.
- Service flows can be deployed to the CICS Service Flow Runtime without changing applications or business processes at all. Typically, all the integration work takes place in the CICS Service Flow Runtime.

Server adapters

A server adapter is a component of a service flow that is invoked during request processing. Depending on what is modeled, the service flow can use server adapters for CICS and IMS applications, transactions,

WebSphere MQ-enabled applications, Web services, and custom programs. If modeled, the FEPI server adapter can perform screen navigation.

In addition to the supported types of server adapters, you can initiate custom programs using an **EXEC CICS LINK** command to augment the functions of a service flow. If you want to include behavior in your service flow that you cannot directly model in service flow project tools, you can write a custom program to perform the function you require and then model a link to it in the flow. The mechanism that invokes the custom program is the same as the mechanism that initiates a CICS application using a program link.

The CICS Service Flow Runtime environment supports the following types of adapters:

- "Front End Programming Interface (FEPI) server adapter" on page 6
- "Link3270 server adapter" on page 7
- "Program link server adapter" on page 7
- "Queue server adapter" on page 8
- "Web service server adapter" on page 8

Front End Programming Interface (FEPI) server adapter

The FEPI server adapter performs screen navigation using the FEPI support in CICS. You must set up FEPI correctly in the CICS region to run this type of server adapter.

service flow project tools has a tool that conducts an interactive 3270 request and reply dialog with CICS and IMS applications. This tool can also perform screen recognition. Using FEPI, it sends requests to and receives replies from any CICS or IMS application with a 3270 data stream that is intended for any size of terminal up to the SLU2 Model 5 (27 rows by 132 columns). You can model and deploy service flows that use the following 3270 screen sizes:

- Model 2 screen size of 24 x 80
- Model 3 screen size of 32 x 80
- Model 4 screen size of 43 x 80
- Model 5 screen size of 27 x 132

Screen sizes that are larger than Model 5 are not supported.

When you model screen navigation in a service flow, a FEPI server adapter is generated to perform the following functions:

1. Begin the FEPI session.
2. Parse screens sent by the CICS or IMS application.
3. Identify the screen and its fields, attributes, and data.
4. Construct and send an appropriate reply, based on the modeling and on simple business logic.
5. Handle the next screen by parsing, identifying, and constructing a reply or keystroke.
6. Manage state information about the status of logical units.
7. End the FEPI session.

The buffer in a single send or receive must not be greater than 25 000 bytes.

You also have control over whether to log off the terminal or continue using the same terminal and screen data in another FEPI service flow. You have these options:

- Force
- Hold
- Leave assigned
- Pass
- Release

service flow project tools saves the option in the service flow properties file when you generate the service flow for deployment. This option is also defined in the service flow repository file as part of the deployment process. During request processing, the navigation manager or flow navigator uses the value of this option to correctly process the FEPI server adapter and, if required, maintain the buffer and connection for another FEPI server adapter to use.

Link3270 server adapter

The Link3270 server adapter performs screen navigation using the CICS Link3270 bridge mechanism. You must set up the Link3270 bridge correctly in the CICS region to run this type of server adapter.

The service flow project tools contains the 3270 emulation and navigation logic for 3270 application screens that use BMS maps or 3270 data streams. The 3270 data stream is intended for any size of terminal up to the SLU2 Model 5 (27 rows by 132 columns).

Link3270 server adapter logic is developed from the point of view of an end user sitting at a 3270 terminal. That is, the server adapter *sees* the business response data of the target 3270 application that the 3270 terminal user would see on the screen. Using either the screen buffer or application data structure (ADS) from the target application program, the Link3270 server adapter identifies the screen and its attributes and constructs an appropriate reply.

The generated server adapter can be the only one in a simple service flow or it can be one of many server adapters in a complex service flow. It performs the following processing:

1. Initiates the target application using the Link3270 bridge.
2. Parses the screens or application data structures sent by the application.
3. Identifies the screen and its fields, attributes, and data.
4. Constructs and sends an appropriate reply, based on the modeling and on simple business logic.
5. Handles the next screen or application data structure from the application.
6. Closes the Link3270 bridge session.

The Link3270 server adapter supports the following CICS API commands when communicating with the target application:

- **SEND MAP**
- **RECEIVE MAP**
- **SEND**
- **RECEIVE**
- **CONVERSE**

Program link server adapter

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

The program link server adapter is called DFHMASDP and is invoked for all programming links that are modeled in the service flow. These links can target either applications in the same CICS region as the CICS Service Flow Runtime environment or in another CICS region. If the target application is remote, DFHMASDP performs a distributed programming link (DPL) to access the remote application. Because DFHMASDP is handling all of the programming links, it is always invoked by a flow navigator and runs by default under the CMAS transaction. You can override the transaction that runs DFHMASDP in service flow project tools.

DFHMASDP supports linking to target applications with either a COMMAREA or using a channel and containers, depending on what was modeled in the service flow. DFHMASDP creates the COMMAREA or channel and containers, links to the application, and then waits for a response before returning to the flow navigator.

Restrictions

The maximum COMMAREA length that can be passed from this server adapter to the target application is 32 767 bytes. 267 bytes are taken up by the message header and the remaining 32 500 bytes can be used for application data. To overcome this limit, use a channel and containers instead. A container can hold up to 2 GB of data.

If you want to pass the data using a channel and containers, you can send or receive only up to 999 containers on a channel. The following container names are reserved for use by CICS Service Flow Runtime:

- ADAPTER.PROCESS
- ADAPTER.ERROR
- COMMAND.STATUS
- COMMAND.INPUT
- COMMAND.OUTPUT
- DPL.DATA

Queue server adapter

The queue server adapter handles the requests and responses for WebSphere MQ-enabled applications. To use the DFHMASCQ server adapter, you must configure the CICS region to use the CICS-WebSphere MQ adapter.

The Queue server adapter is called DFHMASCQ and runs under the CMAU transaction as a child activity of the flow navigator. Any service flow that includes a request to WebSphere MQ adheres to a complex deployment pattern and is therefore always invoked by the flow navigator.

The flow navigator passes several BTS data-containers to DFHMASCQ that define the type of interaction that is required during request processing and the data that should be sent to the target WebSphere MQ-enabled application. DFHMASCQ uses the input from these data-containers and the service flow data-containers to perform its request processing.

DFHMASCQ can issue **MQGET** or **MQPUT** commands, depending on what is modeled in the service flow.

To use the DFHMASCQ server adapter, you must have the required WebSphere MQ definitions configured in the CICS region.

Web service server adapter

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

The Web service server adapter is called DFHMASWS and runs by default under the CMAO transaction. You can override the transaction that runs DFHMASWS in the service flow project tools. DFHMASWS handles all outbound Web service requests, so it is always invoked by the flow navigator. DFHMASWS sends a Web service request by issuing the **EXEC CICS INVOKE WEBSERVICE** command. This request is processed by message handlers in a requester mode pipeline and CICS sends a SOAP message to the designated Web service provider. DFHMASWS waits for a response from the Web service provider and then returns to the flow navigator.

To support the Web service server adapter, the following CICS resources are required:

PIPELINE
: A PIPELINE resource definition is used when a CICS application is in the role of a Web service provider or requester. It provides information about the message handler programs that act on a service request and on the response. The PIPELINE specifies the name of an zFS file that contains an XML description of the message handlers and their configuration. This file is called the pipeline configuration file.

WEBSERVICE
A WEBSERVICE resource defines aspects of the runtime environment for a CICS application program deployed in a Web services setting. It defines the pipeline that the Web service will use, the location of the Web service binding file, and the Web service description (WSDL).

To enable you to quickly deploy adapter services that contain Web service requests, these resources are created during the post installation procedures. A sample requester pipeline called DFHMASFR is the default pipeline that is used by service flow project tools. It contains the basic handlers that are required to process an outbound Web service request and an inbound response message from a Web service provider.

Do not configure this pipeline to include additional handlers. DFHMASFR might be updated when APARs are applied or the product is reinstalled, causing you to lose your configuration changes. To perform additional processing on Web service requests in the pipeline:

- Use an existing CICS requester mode pipeline when you are modeling the flow.
- Create a new pipeline in CICS to handle Web service requests from adapter services and add your own handlers using the pipeline configuration file. Change the default pipeline to the new pipeline when you are modeling the flow.

Sequence of tasks for using CICS Service Flow Runtime and service flow project tools

The following sequence lists the tasks that you typically perform when using service flow project tools and CICS Service Flow Runtime.

About this task

Procedure

1. At build time:
   a) Plan and design your implementation.
   b) Understand, design, and implement the service requester.
   c) Create the service flow using the service flow project tools.
      You will probably include the following tasks:
      i) Capturing existing EIS interfaces.
      ii) Modeling a newly composed business service using these interfaces as a service flow.
      iii) Generating the service flow runtime code.

   For detailed information on the task flow for using the service flow project tools, see the applicable sections of the help in the IBM Developer for Z tooling.
2. Prepare your environment to install CICS Service Flow Runtime:
   a) Ensure that your site meets the programming prerequisites and space and storage requirements necessary to support the CICS Service Flow Runtime.
   b) Unload the CICS Service Flow Runtime modules from the media onto your z/OS server.
   c) Set security and authentication parameters.
   d) Perform the necessary configuration tasks to correctly set up CICS Service Flow Runtime.
3. At run time:
   a) Deploy service flows in the runtime development region on the z/OS server.
      This step assumes that, if required, FEPI, WebSphere MQ, MRO, ISC, IRC, and CICS transactions and custom programs, associated software and systems are already operational and available.
   b) Monitor performance of the CICS Service Flow Runtime.
   c) Address any problems.

d) Deploy the service flows into production.

Service flow runtime terminology

activity
One part of a process managed by CICS business transaction services. Activities implement the business logic. Typically, an activity is part of a business transaction and is executed by a normal CICS transaction responding to CICS BTS events.

basic mapping support (BMS)
An interface between CICS and application programs that formats input and output display data and routes multiple-page output messages without regard for control characters used by various terminals.

business transaction
A self-contained business function; for example, the booking of an airline ticket. Traditionally, in CICS a business transaction might be implemented as multiple user transactions; the booking of the airline ticket might be undertaken by transactions that inquire about availability, reserve the seat, deal with payment, and print the ticket. Using BTS, a business transaction might be implemented as multiple activities.

Business Transaction Services (BTS)
An application programming interface and set of services for implementing complex business transactions in CICS.

CICS-supplied interface
An interface that is used by a controlling application to initiate a CICS program. An application can use one of three interfaces, ECI, EXCI, and EXEC CICS LINK, that are supplied by CICS.

compensation
The act of modifying the effects of a completed activity. The application designer decides how this is implemented, but it often involves the undoing, or reversing, of the actions that the activity took.

data-container
A named area of storage, maintained by BTS, and used to pass data between activities or between different invocations of the same activity. Each data-container is associated with an activity; it is identified by its name and by the activity for which is it a container. An activity can have any number of containers provided that they all have different names.

deployment pattern
A well-defined usage pattern that describes how a service will run in the target environment. Service flows can comply with a set of simple and complex deployment patterns.

enterprise information system (EIS)
The applications that comprise an enterprise's existing system for handling company-wide information. An enterprise information system offers a well-defined set of services that are exposed as local or remote interfaces or both.

enterprise information system interface
The data source in an enterprise information system; for example, 5250 and 3270 screens, COBOL record descriptions and transactions. Using service flow project tools, developers model and compose these interfaces into a more SOA compliant programming interface, enabling the enterprise to transform or adapt to a new set of operations and methods that move the application towards a service oriented architecture.

Link3270 bridge mechanism
A facility in CICS that provides a simplified interface using LINK, ECI, and EXCI. An application uses the Link3270 bridge to run 3270 transactions by linking to the DFHL3270 program in the router region and passing a COMMAREA that identifies the transaction to be run and contains the data used by the user application. If the target application uses BMS, the reply is presented in the form of an application data structure (ADS), another name for the symbolic map that is generated by the BMS macros used to define the mapping of the 3270 terminal screen.

persistence
A characteristic of data that is maintained across session boundaries, or of an object that continues to exist after the execution of the program or process that created it, usually in nonvolatile storage such as a database system.

process
In BTS, a collection of one or more activities. A process is the largest unit with which CICS business transaction services can work. It has a unique name by which it can be referenced and invoked. Typically, a process is an instance of a business transaction.

root activity
The activity at the top of the activity tree; that is, it has no parent activity. The root activity is the control program for a business transaction that represents the start and the end of the process. It initiates and controls a set of child activities.

runtime environment
The CICS region where the CICS Service Flow Runtime is installed and where a developer can deploy a service flow.

screen
In its native state, a screen represents the user interface to a 3270 or 5250 application on a host system. A single host application can contain many screens, each of which has a purpose in the context of the application. Screens contain both text and control or formatting functions, and traditionally display on 3270 or 5250 terminals.

service flow
A graphical representation of the sequence of activities performed in accordance with the business processes of an enterprise. A service flow consists of a graph of nodes, with defined entry and exit points where each node represents invoking a service operation, controlling the flow of the sequence, or performing reusable business logic.

Service-oriented architecture (SOA)
An architecture pattern that describes at a conceptual level the structure of a software system in terms of its components and the services they provide, without regard for the underlying implementation of these components, services, and the connections between components.

transaction
A unit of processing consisting of one or more application programs, affecting one or more objects, that is initiated by a single request.

transform
A change in the structure and values of data from one form to another. At build time, a developer can use service flow project tools to transform existing interfaces in an EIS to enable participation of EIS applications in an SOA.

Chapter 2. What's new

CICS Service Flow Runtime Version 3.2 delivers a set of capabilities that support the enhanced service flow project tools to increase business flexibility, and improves the management of deployed service flows. CICS Service Flow Runtime can run on CICS Transaction Server Version 3.2, Version 4.1, and V5.

Support for Rational Developer for IBM z Systems

CICS Service Flow Runtime 3.2 supports IBM Developer for Z version 7.1.

Many of the enhancements to CICS Service Flow Runtime are in support of the enhancements to IBM Developer for Z version 7.1. Therefore, you cannot model and deploy flows into this level of the runtime environment using an earlier version of the tooling. However, you can regenerate any existing flows and redeploy them into this release of the runtime environment without making any changes.

You can also use IBM Developer for Z version 7.1 to deploy service flows into CICS Service Flow Runtime 3.1.

Changes to the installation

The postinstallation procedures that are provided to correctly set up CICS Service Flow Runtime have changed.

The postinstallation procedures now require you to define a new PLT program called DFHMAINS. This program recovers any deployed service flows if the CICS region is restarted.

Because of the changes in the server adapters in this release, the installation verification procedures and simulator program have been removed.

Enhancements to deploying and managing service flows

The process of deploying service flows now includes the flexibility to define resources dynamically using a new service flow properties file. You can now manage the deployed service flows using a management transaction.

service flow project tools now generates a service flow file that describes the modeled service flow. The service flow file contains the definition of the server adapters, programs, and transactions that are required to run the service flow in CICS. When you deploy this file into the runtime environment, it replaces the properties file update job that existed in previous releases.

The RDO job that updates the CSD is now optionally generated and is not required to define the service flows in CICS. Instead of updating the CSD, you can use the service flow file to create the resource definitions. This option is set in service flow project tools. However, the definitions for the service flow do not persist across a cold start of the CICS region.

The service flow repository file DFHMAASF

A new service flow repository file called DFHMAASF replaces the properties file. The new file contains a record for every service flow that is deployed. When a request message is received by the CICS SFR program interface, DFHMADPL, to invoke a service flow, DFHMADPL reads the service flow repository file and puts the information that is defined for the service flow into BTS data-containers. The flow navigator and server adapters uses these BTS data-containers in their processing.

The flow management transaction CMAN

The supplied transaction CMAN enables you to browse the definitions for each service flow in the service flow repository file. You can use the transaction to install, enable, disable, and delete service flows.

New processing mode

An additional processing mode has been added to CICS Service Flow Runtime. The link mode give you more flexibility when deciding how to run service flows in CICS.

Link mode enables a synchronous service flow to run in a single unit of work. This improves performance by using fewer tasks during request processing, but 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.

Enhancements to server 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 now support screen recognition, as modeled in service flow project tools. In addition, the restriction on the screen sizes that the FEPI adapter supports has been lifted. In previous releases, you were restricted to a maximum screen size of 24 rows x 80 columns (model 2). You can now model and deploy service flows that use the following 3270 screen sizes:

- Model 2 screen size of 24 x 80
- Model 3 screen size of 32 x 80
- Model 4 screen size of 43 x 80
- Model 5 screen size of 27 x 132

Screen sizes that are larger than Model 5 are not supported.

To support the larger screen sizes, the buffer size that is specified in the target interaction file has increased to 8,192 bytes. The maximum buffer that can be sent to or from the terminal is 25,000 bytes.

Link3270 server adapter

The Link3270 server adapter now supports target applications that use a 3270 data stream to send information during service flow processing. Structured fields, with the exception of QUERY PARTITION are not supported. The 3270 data stream supports the following screen sizes:

- Model 2 screen size of 24 x 80
- Model 3 screen size of 32 x 80
- Model 4 screen size of 43 x 80
- Model 5 screen size of 27 x 132

Screen sizes that are larger than Model 5 are not supported.

The Link3270 server adapter now supports the following CICS API commands:

- SEND
- RECEIVE
- 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.

DFHMASDP still runs under transaction CMAS by default, but you can override this transaction to use your own in service flow project tools. 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-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 project tools.

DFHMASCQ still runs under transaction CMAU, but uses new BTS data-containers to perform its processing.

Web service server adapter

The Web service server adapter DFHMASWS still runs under transaction CMAO by default, but you can override this transaction to use your own in service flow project tools.

Changes to error handling

The error message handling has been enhanced and the format of every message that is issued has changed.

- The prefix of all the messages has changed from CIA to DFHMA.
- Every message now contains a standard set of information that you can use to diagnose problems in the runtime environment.

The messages and codes are now written to a transient data queue called CMAC rather than the DFHMAERF error file. CMAC is an alias of CSMT, so that any message issued by CICS SFR appears with the CICS messages. The DFHMAERF error file and the utility DFHMAEUP, which was used to dump and format the error file, have been removed in this release.

If you have written a program to capture messages with a CIA prefix, or if you have written your own program to look at the dump file, you must change your program accordingly.

New messages are issued to support the enhancements in this release; for example, the CMAN transaction messages and the service flow recovery messages.

New trace points are available so that, when you enable user tracing, an exception trace is issued at the same time as the error message. The exception trace data contains additional information about why the error occurred.

Chapter 3. Planning

You must perform a number of activities when planning to use the service flow project tools with CICS Service Flow Runtime

About this task

Using service flow project tools to transform existing applications on an enterprise information system requires the participation of project team members, including host application developers, in the following activities:

Procedure

1. Discovery
2. Planning
3. Developing
4. Deploying

Results

You can find out more information about developing and deploying using service flow project tools in the IBM Developer for Z information center. It contains information on how to develop new services using existing applications and how to deploy the services to CICS.

Discovery phase of an application transformation project

The following activities apply to the discovery phase of an application transformation project.

Procedure

- Analyze the current business climate that drives the enterprise to make changes to existing processes and IT systems.
- Evaluate how existing applications and their interfaces work currently to service the business, with the goal of identifying unnecessary manual intervention and problems.
- Understand in detail the applications and systems to be transformed.

Planning phase of an application transformation project

The following activities apply to the planning phase of an application transformation project.

Procedure

1. Plan and design your implementation.
   - Understand and design business transactions.
   - Decide if and how to implement recoverability at the request message level.
   - Decide how to implement security.
   - Determine the BTS configuration that is necessary to support service flow request processing.
2. Determine whether you require BTS auditing.
3. Develop a test that proves the run time works as expected.

Deployment patterns

Developers use service flow project tools to define a combination of elements or characteristics that describe how the service flow runs in the CICS Service Flow Runtime environment to process requests.

Deployment patterns vary depending on the characteristics of the service flow and the nature of the request. For example, a service flow that models a complex business transaction might require access to multiple target applications and might result in data being updated. While a service flow that models a simple business transaction might require access to only a single target application and result in no data being updated. The differences between simple and complex service flows mean that different processing patterns are used at run time.

Single connector *simple* patterns

A single connector pattern consists of a single server adapter program as modeled in service flow project tools to compose a service flow. The server adapter program can consist of one or more interactions with multiple target systems, but does not require a flow navigator to manage request processing. This pattern supports simple screen-sequencing if the server adapter type is FEPI or Link3270. Only one server adapter can be generated in a single connector pattern.

The single connector pattern does not support WebSphere MQ server adapters, the program link server adapter, and the Web service server adapter.

There are two single connector patterns:

Nonpersistent single connector pattern
The term *nonpersistent* means that no record is written to the BTS repository data set to reserve the name of the BTS process. If a pattern is nonpersistent, the CICS SFR interface program uses the BTS NOCHECK option on the **BTS DEFINE PROCESS** command. Using this option improves BTS performance by removing the need to write to the repository and its associated logging. It also requires little if any BTS configuration.

Persistent single connector pattern
The term *persistent* means that a record is written to the BTS repository data set to reserve the name of the BTS process. In this pattern, the CICS SFR interface program does not use the BTS NOCHECK option on the **BTS DEFINE PROCESS** command. Omitting this option ensures that the context of the service flow (the request and reply data, as well as an intermediate state data) is persistent through a failure.

Aggregate connector *complex* patterns

An aggregate connector pattern consists of multiple server adapter programs as modeled in service flow project tools to compose a service flow. Each server adapter program can consist of one or more interactions with multiple target systems. This pattern also includes the use of a generated and deployed flow navigator to mediate and control the progress of the service flow. During service flow request processing, multiple server adapters can run without requiring action by the service requester.

There are two aggregate connector patterns:

Nonpersistent single connector pattern
The term *nonpersistent* means that no record is written to the BTS repository data set to reserve the name of the BTS process. If a pattern is nonpersistent, the CICS SFR interface program uses the BTS NOCHECK option on the **BTS DEFINE PROCESS** command. Using this option improves BTS performance by removing the need to write to the repository and its associated logging. It also requires little if any BTS configuration.

Persistent single connector pattern
The term *persistent* means that a record is written to the BTS repository data set to reserve the name of the BTS process. In this pattern, the CICS SFR interface program does not use the BTS NOCHECK option on the **BTS DEFINE PROCESS** command. Omitting this option ensures that the context of the service flow (the request and reply data, as well as an intermediate state data) is persistent through a failure.

How the processing mode is selected

The service flow project tools enables a developer to select from five processing modes. These processing modes define how the service requester invokes a service flow and how the service flow is processed, either synchronously or asynchronously, regardless of the deployment pattern associated with the service flow.

The processing modes are as follows:

Asynchronous
The asynchronous processing mode is used only when the service requester is a WebSphere MQ-enabled application. All of the BTS activities run asynchronously.

Link
The service requester invokes a service flow synchronously and all of the BTS activities run within a single unit of work. This is the most efficient form of request processing.

Synchronous
The service requester invokes a service flow synchronously and all of the BTS activities run synchronously. If there is an error, the changes are not rolled back

Synchronous roll back
The service requester invokes a service flow synchronously and all of the BTS activities run synchronously. If there is an error, the changes are rolled back.

Although the developer selects the processing mode when the service flow properties file is generated, the processing mode can be overridden by the service requester.

Considering security

You can implement security in two ways. The service requester can either provide the security or you can implement security in CICS.

About this task

You can implement security in the CICS Service Flow Runtime using one of the following components:

Procedure

- Configure RACF® or another external security manager to handle request messages.

  CICS Service Flow Runtime does not require users to sign on before issuing requests for processing. However, you can specify authentication using RACF to check that the user ID is authorized to invoke a service flow.
- Configure the WebSphere MQ-CICS bridge to provide security for service requesters that are WebSphere MQ-enabled applications.

  The WebSphere MQ-CICS bridge uses the **AUTH** parameter that is passed to the CICS bridge monitor task at startup to establish the authentication level required for the CICS bridge link task.

  a) If you are using FEPI with PassTickets, set the **AUTH** parameter to a value other than LOCAL.

  b) To establish different levels of authentication, initiate multiple bridge monitor tasks with different **AUTH** parameters.

  The WebSphere MQ-CICS bridge checks that the user has the authority to run the CICS bridge link task, based on the user ID and password. The CICS Service Flow Runtime programs and processes run under the user ID and password established for the CICS bridge link task. See Security considerations for using IBM MQ with CICS for information on security while using WebSphere MQ with CICS.

Audit levels

Auditing is a function of BTS that records information about BTS processing. The audit level determines the audit points. The service requester can set auditing in the request message.

About this task

The supported audit levels and their signifiers are as follows:

- Activity (A)
- Full (F)
- Process (P)
- None ( )

For a detailed description of how to create an audit trail for BTS processes and activities, seeDeveloping with the BTS API.

Chapter 4. Installing the CICS Service Flow Runtime

The following tasks outline how to set up the runtime environment to work with CICS.

Procedure

1. Check that you have the prerequisites installed on your z/OS server.
2. Install the CICS Service Flow Runtime modules onto your z/OS server.

   The Program Directory also includes details on space and storage requirements.
3. Complete the postinstallation steps for each CICS region where CICS Service Flow Runtime is installed.
4. Restart the CICS region.

What to do next

The following section describes the prerequisites and the postinstallation steps that you should perform.

The service flow project tools is part of the IBM Developer for Z product and is installed separately on client machines.

Software prerequisites

You must have the following products installed in order to use CICS Service Flow Runtime.

- CICS Transaction Server Version 3.2 or later.

  You must configure CICS Business Transaction Services (BTS) in the target region. The CICS conversion table (DFHCNV) must also be available to the runtime environment. If you want to run Link3270 server adapters, you must have the Link3270 bridge facility correctly configured in your CICS regions. For more information about running Link3270 server adapters in a multiregion environment, see “Managing shared temporary storage queues in a multiregion environment” on page 72.

  If you want to use CICS Service Flow Runtime with CICS Transaction Server Version 4.1, you must apply APAR PK75711.
- IBM Enterprise COBOL for z/OS and OS/390® Version 3 Release 1 or later.
- z/OS V1R7.0 Language Environment® or later.
- MQSeries® for OS/390, Version 2.1 or later or WebSphere MQ for z/OS Version 5.2 or later. WebSphere MQ is required if you want to use the following functions:
  - Asynchronous processing
  - Queue server adapter
- If you want to use FEPI PassTickets you must install z/OS Security Server (RACF) Version 2.5, including PTFs UW91119 and UW91120, and PTF UW90545 for APAR OW35612. You can use another type of external security manager provided that it supports PassTickets.

Performing postinstallation tasks

You perform postinstallation tasks to customize and test the CICS Service Flow Runtime software that is installed on your z/OS server.

About this task

Sample JCL is provided in the .SCIZSAMP library to help you quickly perform the necessary set up tasks.

This section contains the following information:

Procedure

1. Instructions on how to customize and run the setup procedures.
2. Instructions on providing the build time templates to the development environment.
3. Optional: Instructions on how to set up data conversion.
4. Optional: Instructions on how to configure the autostart procedure for Link3270 Facility State Cleanup programs.
5. Optional: Instructions on how to add support for BIDI transformations.
6. A list of the supplied jobs that are provided in the .SCIZSAMP library, which you use to define and install the CICS Service Flow Runtime programs, files, and resource definitions to CICS.

What to do next

The tasks are sequential and you should perform them in the order in which they appear in the following topics.

Customizing the setup procedure DFHMAINJ

The DFHMAINJ sample job creates the runtime libraries for CICS Service Flow Runtime. It also copies all of the system libraries to the runtime libraries and customizes them based on a set of parameters that you can edit in the JCL before running the job.

Before you begin

Before you begin to customize the JCL, copy member DFHMAINJ from the .SCIZSAMP library to a new location. Copying the member ensures that any modifications to DFHMAINJ are not overwritten when system maintenance or version upgrades are applied.

About this task

Edit DFHMAINJ as follows:

Procedure

1. Specify a valid job card, change *hlqual* to the high level qualifier of your CICS SFR libraries, and change *syshlq* to the high level qualifier of your SMP/E installation libraries.
2. Provide values for the mandatory parameters and any optional parameters if required.

   The list of parameters is described in “DFHMAINJ parameters” on page 23.
3. Optional: If you are using IBM Enterprise COBOL for z/OS V5.1 create hlqual.SCIZLOAD as a PDSE dataset. Change the JCL in the CREATE step for SCIZLOAD from 'DSNTYPE=PDS' TO 'DSNTYPE=LIBRARY'.
4. Optional: By default, the parameter values in DFHMAINJ are validated before the customization of the libraries take place. If you do not want validation to take place, edit DFHMAINJ to change `validate` to `novalidate` on the DFHMAINR invocation statement in the //REXX step.
5. Submit DFHMAINJ and check the output.

   You will see the following messages in the //SYSTSPRT of the DFHMAINJ job output:

   ```
   DFHMAI1002I SCIZSAMP customization beginning.
   DFHMAI1000I Validation of input parameters is taking place.
   .
   .
   DFHMAI1011I SCIZSAMP customization ended without errors.
   ```

   If any of the parameter values that you have specified has a problem and validation is switched on, the customization of the libraries does not take place. The job output contains one or more DFHMAI prefixed messages that tell you which parameter values are causing the errors.

6. Optional: If you have a BTS repository already installed and defined in the CICS region, you can use it in the runtime environment instead of the new BTS file that CICS SFR creates. Edit the member DFHMASCC in .SCIZSAMP to remove the RDO definition for the BTS file.

   DFHMASET still creates a new BTS file when you run it, but it is not referenced by CICS SFR.

Results

Three runtime libraries are created and the members copied into them:

- .SCIZSAMP, containing JCL, parameter members, and sample jobs
- .SCIZMAC, containing copybooks
- .SCIZLOAD, containing executable members

If the job fails, no customization takes place. When you have validation switched on, if one of the parameter values in the job has a problem, no customization takes place. The job output contains one or more error messages explaining why the customization did not take place. After you have corrected the cause of the error, rerun DFHMAINJ to perform the customization on your runtime libraries.

Example

The customized DFHMAINJ JCL parameters will look similar to the following JCL:

```
*****************************************************************
JOB1        //++++++++ JOB ,CLASS=M,REGION=0M,
JOB2        //         NOTIFY=&SYSUID,MSGCLASS=H
JOB3        //*
*
SHLQ        ANTZ.DFHMA000.INC10
QUAL        WARDABL.ANTZTEST
VOLSER      P2P210
RDOLIST     CICSSFRL
CSDNAME     WARDABL.ZED3.DFHCSD
HLQCICS     CTS310.CICS640
HLQCOBOL    PP.COBOL390.V330
HLQCEE
CEE

WSDIR_REQ   /zfs/wsbind/file/directory/structure/
CONFIG_REQ  /usr/lpp/cicsts/samples/pipelines/\
            \basicsoap11requester.xml
SHELF_REQ   /var/cicsts/
WSDIR_PROV  /zfs/wsbind/file/directory/structure/
CONFIG_PROV /usr/lpp/cicsts/samples/pipelines/\
            \basicsoap11provider.xml
SHELF_PROV  /var/cicsts/
*
*****************************************************************
*                                                               *
* Optional values.                                              *
*                                                               *
*                                                               *
*****************************************************************
PREFIX      TEST
*
/*
//
```

DFHMAINJ parameters

DFHMAINJ has mandatory and optional parameters that you can optionally validate before any customization of the installation libraries takes place.

Mandatory parameters

JOB1

Together with JOB2 and JOB3, JOB1 is used to create the JCL JOB statements for the required jobs in the .SCIZSAMP library. Do not alter the number of + symbols at the beginning of the statement, because these characters are used to substitute the jobname when DFHMAINJ runs.

JOB2
JCL JOB statement continued.

JOB3
JCL JOB statement continued.

SHLQ ***your.smpe.install.hlq***
A 1-35 character length value that is the data set name high-level qualifier of the CICS SFR SMP/E installation libraries. This value must match what you specified in the previous step for **syshlq**.

QUAL ***your.runtime.library.hlq***
A 1-35 character length value that is the data set name high-level qualifier of the runtime libraries. This value must match what you specified in the previous step for **hlqual**.

VOLSER ***vvvvvv***
A 1-6 character value of the volume serial number that you use for data set allocations.

If you use Storage Management Subsystem (SMS) to manage the creation of your data sets, this parameter is ignored when the data sets are allocated.

Acceptable characters:

```
A-Z 0-9 ./_ ( $ # @
```

RDOLIST ***grplist***
The name of the CICS RDO list that contains the CICS SFR group and, optionally, the WebSphere MQ groups. You are recommended to use the list CICSSFRL.

Acceptable characters:

```
A-Z 0-9 $ # @
```

CSDNAME ***your.cics.dfhcsd***
A 1-44 character length value that is the fully qualified name of the CICS DFHCSD file. This file must exist.

HLQCICS ***your.cics.hlq***
A 1-35 character length value that is the high-level qualifier of the CICS libraries.

HLQCOBOL ***your.cobol.hlq***
A 1-35 character length value that is the high-level qualifier of the COBOL runtime libraries.

HLQCEE ***your.language.environment.hlq***
A 1-35 character length value that is the high-level qualifier of the Language Environment runtime libraries.

WSDIR_REQ ***/your/wsdir/requester/***
The fully qualified name of the Web service pickup directory on zFS that contains the Web service binding file and optionally the WSDL for your Web service requester application. The length of the fully qualified directory name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

CONFIG_REQ ***/your/pipeline/configuration/requester_config.xml***
The name and location of the requester mode pipeline configuration file in zFS. For example: `/usr/lpp/cicsts/samples/pipelines/basicsoap11requester.xml`. The pipeline configuration file defines the message handlers that process outbound and inbound Web service requests for your Web service requester application. The length of the fully qualified directory name must not exceed 255 characters and must start with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory and file name are case-sensitive.

SHELF_REQ ***your/shelf/directory/***
The fully qualified name of the directory on zFS that contains subdirectories for the requester mode pipeline configuration files and Web service requester binding files. The length of the fully qualified directory name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

WSDIR_PROV ***/your/wsdir/provider/***
The fully qualified name of the Web service pickup directory on zFS that contains the Web service binding file and optionally the WSDL for your Web service provider application. The length of the fully qualified directory name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

CONFIG_PROV ***your/pipeline/configuration/provider_config.xml***
The name and location of the provider mode pipeline configuration file in zFS. For example: `/usr/lpp/cicsts/samples/pipelines/basicsoap11provider.xml`. The pipeline configuration file defines the message handlers that process inbound and outbound Web service requests for your Web service provider application. The length of the fully qualified directory name must not exceed 255 characters and must start with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory and file name are case-sensitive.

SHELF_PROV ***/your/shelf/directory/***
The fully qualified name of the directory on zFS that contains subdirectories for the provider mode pipeline configuration files and Web service provider binding files. The length of the fully qualified name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

Optional parameters

PREFIX ***your.prefix***
A 1-7 character length value. The JCL jobname is created as a combination of this value and the name of the member that is customized. For example, if you specify `PREFIX  CSFR`, you will rename every jobname in the members of the runtime SCIZSAMP library with CSFR as the first four characters; for example, //DFHMASET becomes //CSFRASET. If you do not specify a value for this parameter, the sample jobnames are the same as the member names.

Acceptable characters:

```
A-Z a-z 0-9
```

The first character of this value must not be a number.

Running the product definition procedure DFHMASET

The JCL program DFHMASET completes the installation for CICS SFR. It compiles programs, creates and initializes all of the necessary files, and creates the required CICS resources.

About this task

The member DFHMASET is in the runtime library .SCIZSAMP and was copied there and customized when you ran DFHMAINJ.

Procedure

1. Run DFHMASET.

   The job executes these steps:

   - Compiles the file initialization program and the record delete program. If you are using IBM Enterprise COBOL for z/OS V5.1 or above, you will need to take the following step to update the supplied JCL :

     Update the JCL in QUAL.**.SCIZSAMP(DFHMAIBP) and (DFHMAXCP) with the additional SYSUT* and SYSMDECK libraries required for IBM Enterprise COBOL for z/OS V5.1 and above. You should increase the JCL parameter REGION size accordingly. 'QUAL' is the high-level qualifier of the runtime libraries that you specified in your DFHMAINJ parameters. For more information see Enterprise COBOL for z/OS documentation library.

   - Creates the following files:

BTS	BTS file
DFHMACOF	FEPI SLU connection file
DFHMALVA	Link Bridge vector log file
DFHMALVB	Link Bridge vector log file
DFHMAL2F	Link Bridge state file
DFHMATIF	FEPI target interaction file

   - Initializes DFHMACOF and creates an alternate index DFHMAC1F for it.
   - Updates the CSD to create the CICS resources.

2. Check the job output.

   All steps must have a return code of 0, except for the compiling of DFHMADCD, DFHMADCI, and DFHMAVUP. A return code of 4, accompanied by error message IGYDS0001-W, is acceptable for these steps.

   If you have specified compiler options `OPTIMIZE(STD)` or `OPTIMIZE(FULL)`, a return code of 4, accompanied by error message IGYOP3091-W, is also acceptable for these steps.

3. Update CICS JCL to specify the runtime library SCIZLOAD in the DFHRPL concatenation.
4. Include the name of the CICS RDO list that contains the CICS SFR group in the **`GRPLIST`** system initialization parameter.

   You used this name for the **`RDOLIST`** parameter in DFHMAINJ.

What to do next

You can optionally run further steps to enable data conversion, configure the autostart procedure for the Link3270 Facility State Cleanup programs, and enable support for BIDI transformation.

Defining the PLT program DFHMAINS

DFHMAINS is a program list table (PLT) program that installs deployed service flows to CICS when the region is restarted.

About this task

To define the PLT program DFHMAINS, perform the following steps:

Procedure

1. Define DFHMAINS to the program list table postinitialization (PLTPI) list after the DFHDELIM statement in the PLT.

   DFHMAINS is a second-phase PLT program.
2. Ensure that the suffix of the PLT matches a value for the **PLTPI** system initialization parameter.
3. You can choose whether DFHMAINS should search the service flow deployment directory on CICS start up by adding a statement to the **INITPARM** system initialization parameter.
   - If you want DFHMAINS to search the service flow deployment directory, specify the following value:

     ```
     INITPARM=(DFHMAINS='/zFS/repository_directory/')
     ```

     where */zFS/repository_directory/* is the location of the deployed service flow files. The service flow files are created by service flow project tools and deployed into this location.

     If you restart the CICS region, the service flows are redefined to CICS by DFHMAINS as part of the system initialization process.
   - If you do not want DFHMAINS to search the service flow deployment directory, specify the following value:

     ```
     INITPARM=(DFHMAINS='NO_SFR_INST')
     ```

     If you restart the CICS region, the service flows are not redefined to CICS by DFHMAINS as part of the system initialization process.

Defining the security of transactions

If you have implemented security in your CICS region using an external security manager such as RACF, you must authorize the category 2 transactions that are used by CICS SFR.

About this task

You authorize the transactions in RACF by creating a profile in the transaction attach class. To create a RACF profile for the CICS SFR-supplied transactions you must do the following:

Procedure

1. Decide which user IDs should own and have access to the CICS SFR category 2 transactions.
2. Issue the following command:

   ```
   RDEFINE GCICSTRN CICSSFR UACC(NONE)
            ADDMEM(CMAD,CMAF,CMAK,CMAN,CMAO,CMAS,CMAU,CMIT)
            NOTIFY(security_admin_userid)
            OWNER(userid or groupid)
   PERMIT   CICSSFR CLASS(GCICSTRN) ID(sfrusr1,...,sfrusrz) ACCESS(READ)
   ```

   adding in suitable user IDs to authorize users to run the transactions.

   You can use the sample program, DFH$CAT2, as a basis for your own program. DFH$CAT2 is in the CICS samples library, SDFHSAMP.

Copying the build-time templates

Build-time templates are customized automatically for you during the setup of CICS SFR. They are required to deploy service flows in your development environment.

Before you begin

The build-time templates are listed in Table 1 on page 28. These templates are customized when you run the set up procedure DFHMAINJ. Copy the customized versions to every client machine intending to generate and deploy service flows in CICS SFR.

About this task

Use service flow project tools to copy the templates:

Procedure

1. Select **Window** > **Preferences** in service flow project tools.
2. Select **Enterprise Service Tools** > **Service Flow projects** > **JCL Templates**.
3. Select the **Import from Host** button.
4. Expand the host connection and browse the SCIZSAMP data set that contains the customized JCL templates.
5. Select all of the JCL templates from the list.

   service flow project tools installs only the templates that it requires.

Example

Table 1. CICS Service Flow Runtime build time templates

Name	Description
DFHMAXCJ	JCL to compile deployed server adapters
DFHMAXCP	Procedure to compile CICS programs
DFHMAXRD	JCL to define the generated service flow programs and transactions to CICS
DFHMAXRG	CICS resource Group add statement
DFHMAXRP	CICS resource PROGRAM define statement
DFHMAXRR	CICS resource PROCESSTYPE define statement
DFHMAXRT	CICS resource TRANSACTION define statement

Setting up data conversion

If you are using a synchronous interface to invoke the CICS Service Flow Runtime, such as ECI, EXCI, or DPL, you might have to perform data conversion in the runtime routing region, using a customized version of the standard CICS conversion table DFHCNV.

About this task

The customized CICS conversion table (DFHCNV) must specify an entry for the CICS SFR interface program DFHMADPL.

To implement data conversion as described above, perform one of the following options:

Procedure

- Assemble and link-edit the CICS Service Flow Runtime conversion template, using the CICS-supplied procedure DFHAUPLE, to create a load module in the required CICS load library.

  The load library is either *hlq*.SDFHLOAD or *hlq*.SDFHAUTH, which you must specify by the **NAME** parameter of the DFHAUPLE procedure.

  The conversion templates provide the CICS Service Flow Runtime message header structures and layout offsets to include the binary field conversions.
- Use the conversion table supplied with CICS Service Flow Runtime to create a load module in a load library other than a CICS load library.

  The sample conversion table DFHMAXCV is in the samples library .SCIZSAMP.

  If you choose this option, you must add this load library to the CICS DFHRPL or dynamic LIBRARY concatenation. The load library must be higher in the search order than either of the CICS load libraries as specified above.

  The conversion program, DFHCCNV, uses the first conversion table, DFHCNV, to perform conversion.

Configuring the autostart procedure for the Link3270 facility state cleanup programs

The CICS Service Flow Runtime Link3270 facility state cleanup has two programs: DFHMALSC and DFHMALFC.

About this task

DFHMALSC cleans up temporary storage queues (TSQs) for simple, nonpersistent Link3270 service flows.

DFHMALFC cleans up the VSAM file DFHMAL2F for the following types of Link3270 service flows:

- Persistent and nonpersistent complex service flows
- Persistent simple service flows

To start the cleanup programs in the final stages of CICS initialization, use the program list table (PLT):

Procedure

1. Define a PLT, specifying which program you want to execute at CICS startup. You can specify both if required.
   For example:

```
DFHPLTPI TITLE 'DFHPLTPI - PROGRAM LIST TABLE STARTUP '
               DFHPLT TYPE=INITIAL,SUFFIX=D1
*
    *-------------------------------------------------------------------------------------------------*
    *  PHASE 2 PROGRAMS FOLLOW                                                                        *
    *-------------------------------------------------------------------------------------------------*
                 DFHPLT TYPE=ENTRY,                                                              x
                       PROGRAM=DFHDELIM
    *
    *-------------------------------------------------------------------------------------------------*
    *  PHASE 3 PROGRAMS FOLLOW DFHDELIM                                                               *
    *-------------------------------------------------------------------------------------------------
*
    *-------------------------------------------------------------------------------------------------*
    *  CICS SFR LINK3270 FACILITY STATE CLEANUP PROGRAM                                               *
    *-------------------------------------------------------------------------------------------------*
              DFHPLT TYPE=ENTRY,                                                                 x
                       PROGRAM=DFHMALSC
          DFHPLT TYPE=FINAL
          END
```

   For programming information about writing PLT programs, see Developing system programs. For information on defining a PLT, see Resources.
2. Define system initialization parameters **PLTPI** and **INITPARM** in the system initialization table (SIT).

The **PLTPI** parameter specifies the suffix of the program list table, which contains the entry for DFHMALSC or DFHMALFC. The **PLTPI** parameter definition in the SIT for the above example would be PLTPI=D1. The **INITPARM** parameter is used to pass parameters to the program, in this example DFHMALSC, which is executed in the final stages of system initialization. The format of the parameter in the SIT is:

```
INITPARM=(DFHMALSC='SI=300')
```

where SI=*nnnnn* is a numeric value in seconds and whose value can range from 300 to 99999 seconds. This value is converted to hhmmss and indicates the start interval for subsequent task starts of program DFHMALSC with transid CMAK.

Setting too low a value for the **SI** parameter can affect the performance of the CICS region. To avoid affecting performance, if you set the parameter to less than 300 seconds, the parameter value is ignored and reset to the minimum value of 300 seconds. For more information about the system initialization parameters, see System initialization parameter descriptions and summary.

What to do next

For a description of cleanup processing on TSQs see “Facility state cleanup processing — TSQ” on page 71. For a description of cleanup processing on the VSAM file, see “Facility state cleanup processing — VSAM” on page 72

Adding support for BIDI transformation in service flows

Optionally, you can configure service flows to support BIDI transformation in the service flow project tools. If these flows are required to run in CICS SFR, you must enable the runtime environment to support them.

About this task

IBM Developer for Z provides the BIDI module, FEJBTRX. IBM Developer for Z includes instructions on how to move the modules to your CICS region. To add support for BIDI transformation in service flows:

Procedure

1. Add the module to the DFHRPL concatenation of the JCL that is used to start your CICS region.
2. If you do not autoinstall programs in your CICS region, update the CICS CSD to include the definitions for the module. Use the **CEDA DEFINE** command to define the module to CICS and then install the definition.

Results

The flow navigator uses a BIDI transformation template called DFHMABID to call the specified module when a BIDI transformation is required in the service flow.

If a BIDI transformation fails, the flow navigator issues a DFHMA08008E error to the CMAC transient data queue. Look at the details in the error message to determine what kind of error has occurred.

CICS Service Flow Runtime samples listing

The DFHMAINJ job copies a number of samples from the .SCIZSAMP library to the runtime libraries that are used by CICS SFR.

The samples that are provided in the .SCIZSAMP library are described in the following table.

Table 2. CICS Service Flow Runtime samples

Name	Description
DFHMABAP	BTS Audit Log dump JCL

Table 2. CICS Service Flow Runtime samples (continued)	
Name	**Description**
DFHMABRP	BTS Repository dump JCL
DFHMADBC	SLU Connection VSAM file alternate index build JCL
DFHMADC1	SLU Connection VSAM file alternate index definition JCL
DFHMADCD	SLU Connection VSAM file initialization program
DFHMADCI	SLU Connection VSAM file initialization program
DFHMADDB	JCL to delete and define the BTS Repository file
DFHMADDC	JCL to delete and define SLU Connection VSAM file
DFHMADDL	JCL to delete and define Link3270 State VSAM file
DFHMADDT	JCL to delete and define Target Interaction VSAM file
DFHMADEA	Link3270 Vector Log file IDCAMS delete
DFHMADEC	SLU Connection file IDCAMS delete
DFHMADEL	Link3270 State file IDCAMS delete
DFHMADET	Target Interaction file IDCAMS delete
DFHMADFA	Link3270 Vector Log file IDCAMS define
DFHMADFC	SLU Connection file IDCAMS define
DFHMADFE	Error file IDCAMS define
DFHMADFL	Link3270 State file IDCAMS define
DFHMADFT	Target Interaction file IDCAMS define
DFHMAINJ	JCL that creates the runtime libraries and contains parameters for customizing those libraries
DFHMAINR	REXX executable that performs customization on runtime library members using the parameter values specified in DFHMAINJ
DFHMAMVD	Link3270 Vector Log file dump JCL
DFHMASET	Creates the set up resources
DFHMASFP	Sample provider mode pipeline that can be used when a flow is modeled as a Web service provider
DFHMASFR	Sample requester mode pipeline that can be used for Web service requests
DFHMAXCV	Template for conversion

Chapter 5. Deploying a service flow

When you generate the runtime code for a service flow in service flow project tools, you perform a series of steps to deploy it to CICS.

About this task

A summary of the deployment steps is as follows:

Procedure

1. Compile the output from service flow project tools.
2. Deploy the service flow to CICS.

   This deployment includes the service flow properties file, RDO JCL, and programs that are required to define the service flow in CICS.
3. Install and enable the service flow, using the supplied flow management transaction, CMAN.

Example

The following figure shows the deployment process.

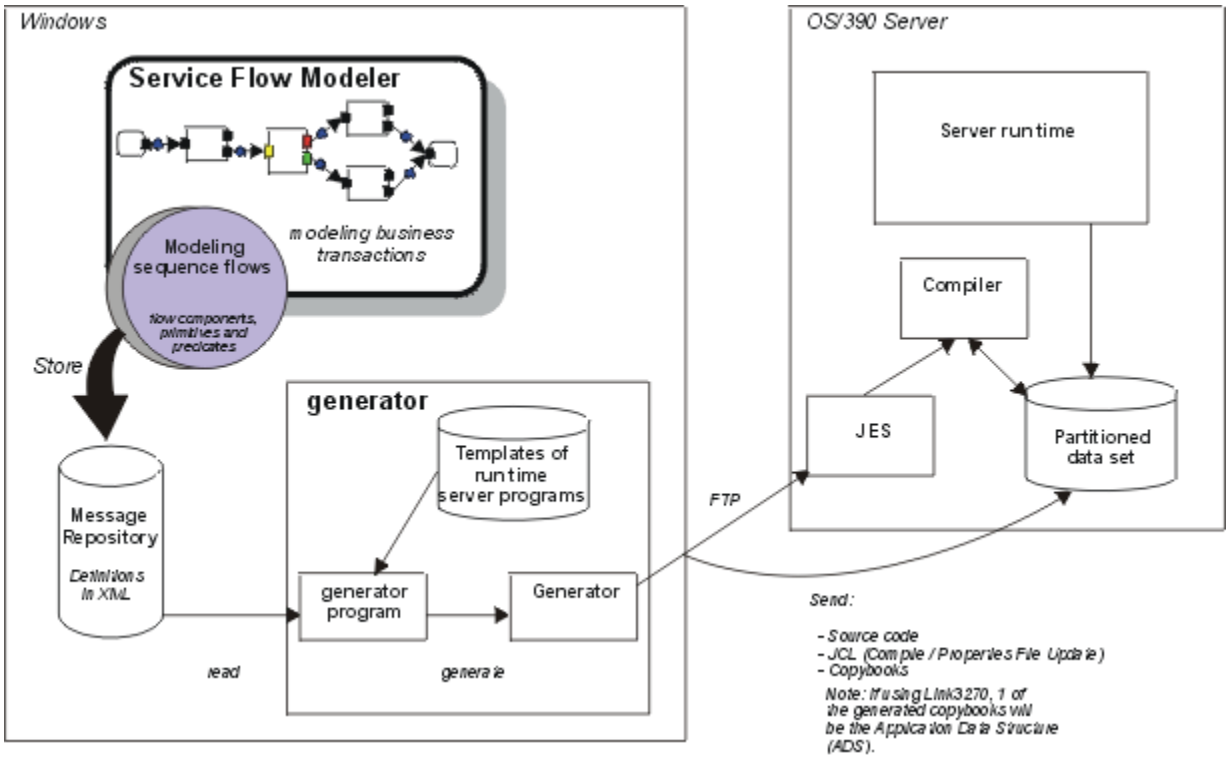


Figure 3. Deploying a service flow from the service flow project tools to the z/OS server

Deploying a new service flow

When you generate the runtime code for a service flow in the tooling, all of the necessary files are created for you to quickly deploy it into CICS. These files include JCL that compiles the required programs, an optional CSD update job that defines the resources to CICS, and a service flow properties file.

Before you begin

You can either run the JCL procedures remotely from service flow project tools or run them directly in MVS. If you want to run them remotely, you must have remote systems configured in the tooling.

About this task

To deploy a new service flow:

Procedure

1. Optional: Copy the generated service flow output from the tooling to the correct location in MVS.

   If you have remote systems configured and select to deploy the service flow remotely, this deployment occurs automatically as part of the generation process.
2. Optional: Run the batch job `#`*jobname*`.jcl`, where *jobname* is the name of the service flow project.

   This file is generated only when you select the appropriate option in service flow project tools.

   The JCL uses a CICS-supplied utility, DFHCSDUP, to update the CSD of the CICS region with the resource definitions for the programs, transactions, and files for your service flow. If a resource is already defined, the job does not replace it.
3. Copy the generated service flow properties file to the zFS directory that contains your service flows.

   This file has a `.sfp` file extension. You can also perform this step using service flow project tools.
4. Run any remaining JCL jobs to compile the programs that are needed by your service flow.
5. Optional: If you selected that the CICS resources should not be installed automatically when you install the service flow, you must install the required CICS resources using the group name.

   The group must belong to the RDO list that was specified in the **RDOLIST** parameter of the postinstallation procedure DFHMAINJ. The default list is CICSSFRL.
6. Optional: If your service flow is enabled as a Web service provider or includes an outbound Web service request, follow these steps:
   a) Ensure that the appropriate PIPELINE resource is enabled in the CICS region to process the Web service requests.

      CICS Service Flow Runtime provides sample requester and provider mode pipelines, and creates the PIPELINE resources for these during the postinstallation steps.
   b) Copy the generated Web service binding file and optionally the WSDL to the pipeline pickup directory in HFS.

      Ensure that you put the files in the correct pipeline pickup directory. If you try to install a Web service binding file for a Web service provider application into a requester mode pipeline or vice versa, the CICS resources do not install in an enabled state. If you copy the WSDL, you can run validation against the Web service at run time to check that the data is being processed correctly.
   c) Run a scan of the pipeline using the **PERFORM PIPELINE SCAN** command.

      This command checks the pickup directory for new files and creates a WEBSERVICE resource based on the Web service binding file. If the Web service is a provider, a URIMAP resource is also created.
   d) Optional: If your service flow is enabled as a Web service provider, and you want to receive Web service requests over HTTP, create a TCPIPSERVICE resource.

What to do next

When you have completed these steps, the service flow is ready for you to install to complete the deployment process.

Installing service flows

Use the CMAN transaction to install newly deployed service flows.

Before you begin

You must deploy the service flows to the deployment directory that is specified for the DFHMAINS program.

About this task

You can install a service flow using either of the following methods:

Procedure

- Enter `CMAN INSTALL`.

  The transaction invokes the DFHMAINS program to install all new service flows and define them in the service flow repository file.
- Enter `CMAN` on your display.

  The transaction displays the flow management main menu where you can perform actions on service flows.

  a) Press PF2 to install newly deployed service flows.

  The DFHMAINS program checks the zFS deployment directory for new service flows and installs them, updating the service flow repository.

  A message confirms whether the installation was successful.

  b) Press Enter to refresh the display.

The service flow repository file

The service flow repository file, DFHMAASF, contains a record for every service flow that is deployed in the runtime environment. The record contains the request name and the properties of the service flow and definitions for each server adapter.

When you deploy a new service flow, the service flow repository file is updated to include all of the information necessary to perform request processing for that service flow. Each service flow record begins with the name of the flow, followed by definitions for the server adapters, programs, and transactions. It can also include additional information:

- The persistence of the request processing
- The processing mode, either synchronous or asynchronous
- The connector type of the service flow, simple or complex

The only server adapters that are not required in this file are the Web services and Program link server adapters.

The service flow repository file is read by the CICS SFR interface program, DFHMADPL, which then creates BTS data-containers for the Navigation manager and server adapters to access during request processing.

You can view the contents of the service flow repository file by using the flow management transaction, CMAN. You cannot share the repository file between CICS regions.

Chapter 6. Invoking a service flow

Invoking a service flow occurs when a service requester passes a *request message* to the CICS SFR interface program DFHMADPL using one of the supported interfaces.

The request message consists of one or more message headers and application data. It contains the parameters and information necessary to start runtime processing and to invoke a service flow that is deployed in the runtime environment. The message headers vary depending on the interface that is used by the service requester, the deployment pattern of the service flow, and the types of server adapter that comprise the service flow.

The service requester can use different interfaces to invoke a service flow, but can invoke only one service flow in a unit of work. All of the interfaces support passing the request message in a CICS communication area (COMMAREA). However, when you use a program link, you can choose to pass the request message to the interface program in a CICS COMMAREA or using a channel and containers. If you use a COMMAREA, the maximum length of the entire request message with all the headers is 33 272 bytes. If you use containers, you can send a simplified message header separately from the application data.

The service requester

A service requester is the application that looks for and invokes or initiates an interaction with a service.

A browser driven by a person or a program without a user interface, for example a Web service, can play the role of a service requester. A service requester issues one or more queries to locate a service and to determine how to communicate with that service.

At run time, a service requester looks for and invokes an interaction with the service flow that has been deployed to the CICS Service Flow Runtime. The following table lists the supported interfaces that a service requester can use to pass in the message header and application data.

Service requester type	Interface used
WebSphere MQ-enabled application	WebSphere MQ-CICS bridge. This product serves as the interface between a WebSphere MQ-enabled service requester and CICS. The request message is passed in a WebSphere MQ message to a WebSphere MQ message queue.
Other applications	A CICS-supplied interface, such as **EXEC CICS LINK**, EXCI or ECI. The IBM CICS Transaction Gateway (CTG) product.

The following applications are examples of service requester:

- WebSphere MQ Integrator
- WebSphere MQ Workflow
- Web service
- Any local or remote application that can initiate a CICS program

What the service requester is responsible for

The service requester is responsible for the following aspects of business transaction processing at run time:

- Managing the overall business flow and compensation.

- Managing business context, complex state, multiple requests and replies, and asynchronous request processing.
- Overseeing the continuation of one logical request through multiple requests, if required.
- Adhering to a published XML message format when creating valid XML request messages.
- Performing data conversion if required. See "Data conversion" on page 43 for information on how to perform data conversion.

Invoking a service flow using a CICS-supplied interface

You can use the External Call Interface (ECI), External CICS Interface (EXCI), or Distributed Program Link (DPL), potentially with CICS Transaction Gateway (CTG) or a distributed version of CICS, to invoke a service flow. When you use a CICS-supplied interface, the request can be processed only in synchronous mode.

About this task

To invoke a service flow using a CICS-supplied interface, follow these steps:

Procedure

1. Select the interface that is appropriate for your service requester.
   a) To use CTG to invoke the service flow, use the ECI interface.

      If CTG is installed on z/OS, the EXCI interface is used.

   b) To use DPL, use the **EXEC CICS LINK** command to link from the service requester to the CICS SFR interface program with a request message.

      If you use DPL, you can use a channel and containers to pass the request message, rather than a COMMAREA.
2. Decide on the data conversion that is required.

   It might be necessary to perform data conversion in the CICS Service Flow Runtime routing region, using a customized version of the standard CICS conversion table DFHCNV. For details, see "Data conversion" on page 43.
3. Decide how you want to send the request message from the service requester to invoke the service flow.

   The service requester can send the message in a COMMAREA or, if you are using a **PROGRAM LINK** command, a channel and set of containers.

   - If you want to use a set of containers, see "Sending the request message in containers" on page 40 for the correct combination and format that you can use in the channel.
   - If you want to use a COMMAREA, see "Sending the request message in a COMMAREA" on page 42 for the required format.

   The service requester then waits for a response if required.
4. When the service flow has completed its processing of the request message, the CICS SFR interface program passes the response message back to the service requester.

   The CICS-supplied interface requires a response, but the request message might not require an application reply.

Invoking a service flow using the CICS-MQ bridge

If the service requester is IBM MQ-enabled, you must use the CICS-MQ bridge to pass the request message to the CICS SFR interface program DFHMADPL.

Before you begin

There are many ways to start the CICS-MQ bridge. This section explains how to start the CICS bridge monitor program using IBM MQ triggering.

To start the CICS bridge monitor program using IBM MQ triggering, define the IBM MQ request queue with TRIGGER, TRIGTYPE(FIRST) INITQ('*initiation queue*') PROCESS('*process*') where:

- The *initiation queue* is the queue on which the CKTI trigger monitor is listening.
- The PROCESS definition *process* must specify `APPLTYPE(CICS) APPLICID(CKBR)`. CKBR is the CICS bridge monitor transaction ID. The **USERDATA** parameter in the PROCESS definition can specify AUTH and `WAIT` options for the CICS bridge monitor program.

About this task

The following steps describe how the run time is invoked in asynchronous mode.

Procedure

1. The service requester sends a request message to the request queue.

   This message causes IBM MQ to send a trigger message to the specified initiation queue.
2. The IBM MQ trigger monitor program starts the CICS bridge monitor task, which is part of the CICS-MQ bridge.
3. The CICS bridge monitor browses the request queue.

   If a message has arrived, the CICS bridge monitor starts the CICS bridge link task, which is part of the CICS-MQ bridge.
4. The CICS bridge link task links to the CICS SFR interface program, passing the request message in a COMMAREA, and waits for control to be returned.

   The request message that is passed to the interface program does not include the IBM MQ header data.

   The interface program defines and runs a BTS process that starts the Navigation Manager, passes the information from the request message in a series of BTS data-containers, and waits for the service flow processing to complete.
5. If a reply message is required, the CICS SFR interface program passes it to the CICS bridge link task in the COMMAREA.

   The reply message contains the application data and the message header.

   - In synchronous processing, if the CICS-MQ bridge header (MQCIH) is not present in the request message, it does not appear in the reply message, except in the case of an error.
   - In asynchronous processing, the CICS-MQ bridge header (MQCIH) structure is always included in the reply message.
6. The CICS bridge link task responds to the service requester using an **MQPUT** command, if the MQMD ReplyToQ and ReplyToQMgr are loaded.

Invoking a service flow from a Web service

If your service flow is deployed as a Web service provider, you can use the existing Web services support in CICS to invoke it.

Before you begin

You must enable the provider PIPELINE resource to process the Web service request and response messages and the WEBSERVICE and URIMAP resources for your Web service provider application. If the request is using HTTP, you must also have an enabled TCPIPSERVICE resource.

About this task

To invoke a service from a Web service, follow these steps:

Procedure

1. Send the SOAP message from the service requester to the Web service provider.

   The SOAP message body contains the request message. The contents vary depending on what you modeled in service flow project tools.

   - By default, the SOAP body includes the request name of the service flow that you want to invoke and any application data.
   - If you select to expose the whole message header, the SOAP body includes the DFHMAH header structure and any application data.

   The provider mode pipeline processes the SOAP message, and the contents of the SOAP body are placed in the DFHWS-DATA container by default. If you select to expose the whole message header in service flow project tools, the DFHMAC-ALLPARMS container is used instead.
2. The application handler in the pipeline passes the appropriate container to the CICS SFR interface program DFHMADPL.
3. DFHMADPL checks the channel for additional containers.

   If you have added additional message handlers to the pipeline to override the contents of DFHWS-DATA with the DFHMAC-SYSPARMV1 or DFHMAC-LNK3270V1 containers, DFHMADPL uses the values in these containers instead.
4. DFHMADPL defines and runs the BTS process for the Navigation Manager to invoke the service flow using the values in the containers.

   - If the request message is successfully processed and a response is required, the interface program DFHMADPL places the response in the DFHWS-DATA or DFHMAC-ALLPARMS container. The pipeline creates a SOAP response message and sends it to the service requester.
   - If an error occurs, the interface program creates a SOAP fault message which is sent to the service requester instead. Details of the error are also placed in the DFHMAC-ERROR container on the channel.

Sending the request message in containers

The service requester can send the request message in a series of containers using a channel when using DPL. You do not have to remodel or redeploy your service flows to use channels and containers.

About this task

To send the request message in containers, follow these steps:

Procedure

1. Decide which set of mandatory containers you want to use to pass the request message.

   You can select one of the following options:

- Pass the request name of the service flow that you want to invoke in the DFHMAC-REQUESTV1 container, and the application data in the DFHMAC-USERDATA container. The request name must match the PROCESSTYPE resource that is defined in CICS SFR for the service flow. When you select this option, the runtime environment processes the request message using default values. These defaults assume that the process type is the same as the request name and that the process name can be uniquely generated by the CICS SFR interface program. You can override these default values using additional containers.
- Pass the request name and application data in the DFHWS-DATA container as a Web service request. Use this option when you are exposing your service flow as a Web service provider application. For details on how to do this, see "Invoking a service flow from a Web service" on page 40.
- Pass the message header and application data in the DFHMAC-ALLPARMS container. This container is primarily provided as a migration aid to help you move from a COMMAREA to channels and containers. Use this container to pass the DFHMAH header to CICS SFR. You can pass the application data after the DFHMAH header in this container or pass it in DFHMAC-USERDATA instead.

2. Optional: If you choose to use the DFHMAC-REQUESTV1 or DFHWS-DATA container, you can use optional containers to override some of the default values associated with the service flow.
   a) To process the request message using a different process type, change the request name, or change the process name using the DFHMAC-SYSPARMV1 container.
      - Specify a different process type to override the PROCESSTYPE resource that is defined in CICS for the service flow. By default, the request name that is passed in DFHMAC-REQUESTV1 matches the PROCESSTYPE resource of the service flow.
      - Specify a different request name to override the value that is passed in the DFHMAC-REQUESTV1 or DFHWS-DATA container.
      - Specify a different process name. This must be a unique identifier. By default, CICS generates a unique identifier when the BTS process is created by the interface program.

      Specifying one of these options overrides the values associated with the service flow when the request message is received.
   b) To pass in a state token to reuse an allocated Link3270 bridge facility, use the DFHMAC-LNK3270V1 container.

      If you do not include this container, the Link3270 server adapter creates a new instance of the Link3270 bridge facility.
3. When you have decided on the correct container combination, use the channel and containers CICS API commands in your service requester application to create the required containers and link to the interface program DFHMADPL.
   For example, you could specify the following commands:

```
* Create base request container
EXEC CICS PUT CONTAINER('DFHMAC-REQUESTV1')CHANNEL(MY-CHANNEL) FROM(…)
* Optionally create user data container
EXEC CICS PUT CONTAINER('DFHMAC-USERDATA') CHANNEL(MY-CHANNEL) FROM(…)
* DPL to server adapter via CICS SFR
EXEC CICS LINK PROGRAM('DFHMADPL') CHANNEL(MY-CHANNEL)
```

   When CICS SFR has processed the request message, DFHMADPL places the response in the appropriate container that holds the application data. For example, if the service requester passed the application data in DFHMAC-USERDATA, DFHMADPL places the response in this container for the service requester to retrieve. It also updates the message header. If an error occurs, DFHMADPL passes the DHFMAC-ERROR container back to the service requester. This container holds the error message and other details that the service requester can interpret. If you used DFHMAC-ALLPARMS, the error is also reported in these containers.
4. Use the channel and containers CICS API commands in your service requester application to retrieve the response.
   For example, you could specify the following commands:

```
* Retrieve error from container
EXEC CICS GET CONTAINER('DFHMAC-ERROR') CHANNEL(MY-CHANNEL) SETPTR(…)
* Retrieve original request data from container
EXEC CICS GET CONTAINER('DFHMAC-REQUESTV1') CHANNEL(MY-CHANNEL) SETPTR(…)
* Retrieve user data from container
EXEC CICS GET CONTAINER('DFHMAC-USERDATA') CHANNEL(MY-CHANNEL) SETPR(…)
```

You must ensure that your service requester can handle any error that is reported in the DFHMAC-ERROR container.

Sending the request message in a COMMAREA

When you use a COMMAREA, the maximum length of the request message with all the headers is 33 272 bytes. Each header in the request message has a field that indicates the data structure or application data format comes next in the request message.

About this task

The headers that you must include in the request message depend on the interface that the service requester is using to invoke the service flow.

Procedure

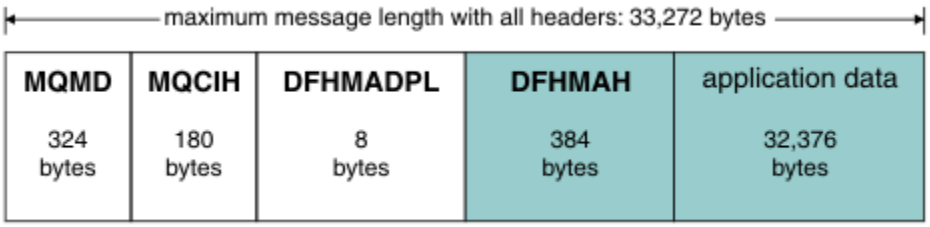
- If you want the service requester to invoke a service flow using a CICS-supplied interface, such as the **EXEC CICS LINK** API command, include the DFHMAH header structure and the application data in the COMMAREA.

  The DFHMAH header describes the structure of the application data.
- If you want the service requester to use the CICS Transaction Gateway (CTG) interface, the request message must contain the 101-byte CTG header, followed by the DFHMAH message header and then the application data.
- If the service requester is a WebSphere MQ-enabled application, the request message must include a WebSphere MQ header (MQMD), the WebSphere MQ CICS Bridge header (MQCIH), the DFHMADPL header, and the message header DFHMAH.

  The MQCIH header is required only when you are implementing password authentication. However, you must always include it to enable compatibility with future releases of CICS Service Flow Runtime. If an error occurs in the WebSphere MQ-CICS bridge, the MQCIH structure is returned to the service requester. This process occurs regardless of whether the structure was sent in the request message.

Example

The following figure illustrates the structure of this request message.



Data conversion

Data conversion refers to the process of changing data from one form of representation to another. A system might perform data conversion when exchanging data with another system that is using a different coded character set identifier (CCSID).

CICS Service Flow Runtime processing does not include data conversion on either the request or reply messages. If data conversion is required, the service requester must implement it. The following sections explain how to implement data conversion when using the supported interfaces from the service requester to the CICS Service Flow Runtime environment.

Data conversion using the IBM MQ interface

The CICS-MQ bridge is an asynchronous interface to the CICS Service Flow Runtime.

If the service requester is an IBM MQ-enabled application, it must use the CICS-MQ bridge when invoking the CICS Service Flow Runtime.

When you require data conversion, either it is the responsibility of the service requester or it can be performed by writing a customized data conversion exit program. For detailed instructions on writing data conversion exits, see the IBM MQ product documentation.

If the service requester has to convert application data between different machine encodings and CCSIDs, the service requester must conform to the IBM MQ data conversion interface. For information on the interface to the data-conversion exit, and the processing performed by the queue manager when data conversion is required, see the IBM MQ product documentation.

Data conversion using a CICS-supplied interface

If you are using a CICS-supplied interface to invoke a service flow, you might have to perform data conversion in the CICS Service Flow Runtime region using a customized version of the standard CICS conversion table, DFHCNV.

You might need a data conversion table if the CICS system is using ISC to communicate with a member of the CICS family that runs on a hardware platform that does not use EBCDIC. The conversion table defines how data is changed from ASCII format at the workstation to EBCDIC format in CICS.

ECI applications use the CICS mirror program to call the CICS data conversion program, DFHCCNV, passing the request and reply messages found in the COMMAREA. DFHCCNV performs the necessary conversion of the COMMAREA as defined in the customized conversion table, DFHCNV. It applies standard conversion to those fields in the conversion templates for which you have not specified nonstandard, user-handled conversion.

If DFHCCNV finds a conversion template in the DFHCNV table that matches the CICS SFR interface program name DFHMADPL, it performs code page translation and data conversion for the COMMAREA associated with the ECI request. A conversion template is a table entry defining fields in a data area to be converted and the conversion method to be applied to each field.

The conversion template must specify the following:

- A table entry for the interface DFHMADPL.
- Entries for each field offset, type of conversion, and field length found in the message header structures DFHMAH.
- All binary fields must specify DATATYP=NUMERIC, if you require ASCII to EBCDIC conversion. You must supply similar conversion entries (DATATYP=NUMERIC) with correct field offsets if the client application data portion of the request and reply messages uses binary data.

See “Conversion template for DFHMADPL” on page 158 for a sample conversion template of DFHMADPL.

For detailed information about the DFHCCNV conversion program, conversion templates, the DFHCNV conversion table, and the syntax of DFHCNV macros, see Configuring CICS interconnectivity and IBM Redbooks: CICS Transaction Gateway V5 The WebSphere Connector for CICS.

You can use other methods to perform Unicode or ASCII to EBCDIC data conversion. For example, you could perform data conversion within a Java™ application. See Developing Java applications and IBM Redbooks: Java Connectors for CICS: Featuring the J2EE Connector Architecture.

For information on how to install and configure DFHCNV, see "Setting up data conversion" on page 28.

Code page conversion

CICS Service Flow Runtime processing does not support native code page conversion for an external combined-client.

A combined-client design requires that a response be received in the form of a 3270 bridge vector, which contains both display and binary data types. Therefore, you are recommended to employ a split-client design where business data exchanged between the clients contains only display data types. The conversion of 3270 Bridge vectors to a custom display format, fixed format COMMAREA, or XML, is performed in the Generic Bridge. The Link3270 bridge in CICS TS 2.2 does provide support for native code page conversion.

Request message containers

Use the request message containers to send the request message header and the application data, with optional parameter values to override defaults in the runtime environment.

Container DFHMAC-ALLPARMS

The service requester uses DFHMAC-ALLPARMS to pass the contents of the DFHMAH header to the CICS SFR interface program, DFHMADPL.

This container is for migration purposes and is an alternative to using the DFHMAC-REQUESTV1 container. It contains the entire DFHMAH header structure in the format that you use to pass a request message in a COMMAREA. The container is mapped by the copybook DFHMAHV, which is in the SCIZMAC library.

You can also include the application data in this container or pass the application data in the DFHMAC-USERDATA container. If you include the application data in this container, you can code it in XML and optionally the DFHMAH header.

If the DFHMAH header and the application data are in XML, the length of the container is limited to 32 760 bytes. If the message header and application data are not in XML, the container can be up to, but not equal to, 16 MB in length.

Container DFHMAC-ERROR

CICS Service Flow Runtime uses DFHMAC-ERROR to return any errors that occur during the processing of the request message to the service requester.

The container is mapped by the copybook DFHMAHEV, which is in the SCIZMAC library.

The following table lists all of the fields that are included in the container, although typically an error does not set all of the fields. A type of field can be as follows:

X
The field contains characters.

FB
The field is a fullword binary.

Field	Length (bytes)	Type	Description
DFHMAHE-RETURNCODE	04	FB	

Field	Length (bytes)	Type	Description
DFHMAHE-COMPCODE	04	FB	
DFHMAHE-MODE	04	FB	
DFHMAHE-SUSPSTATUS	04	FB	The suspend status of the. BTS process
DFHMAHE-ABENDCODE	04	X	The ABEND code. This field is set when the return code is 999.
DFHMAHE-MESSAGE	12	X	
DFHMAHE-FAILED-PROCNAME	36	X	The name of the failed. BTS process
DFHMAHE-FAILED-PROCTYPE	08	X	The type of the BTS. The process type usually process that failedmatches the request name of the service flow.
DFHMAHE-FAILED-TRANID	04	X	The ID of the transaction. that was running when the error occurred
DFHMAHE-FAILED-PROGRAM	08	X	The name of the program. that was active when the error occurred
DFHMAHE-FAILED-NODE	32	X	The name of the node that. was active when the error occurred
DFHMAHE-BRIDGE-RC	04	FB	
DFHMAHE-STATETOKEN	16	X	The state token that. is passed by the Link3270 server adapter to indicate that an allocated Link3270 bridge facility with associated business state data has been stored for subsequent reuse

Container DFHMAC-LNK3270V1

The service requester uses DFHMAC-LNK3270V1 to pass parameter values that are specific to the Link3270 bridge. These parameter values override any defaults in the runtime environment.

The DFHMAC-LNK3270V1 container is mapped by copybook DFHMAHLV, which is in the SCIZMAC library. This optional container can pass the following information:

Field	Length (bytes)	Type	Value
DFHMAH-STATETOKEN	16	Character	A token that is used when an allocated Link3270 bridge facility with state data has been stored for reuse.

Container DFHMAC-REQUESTV1

The service requester uses the DFHMAC-REQUESTV1 container to pass the request name of the service flow to the CICS SFR interface program DFHMADPL.

The request name is eight characters in length and, by default, is the same as the PROCESSTYPE resource for the service flow. When this container is passed to the DFHMADPL program, DFHMADPL uses the default values for the header in request processing. The service requester can override these default values by passing in additional containers. The container is mapped by the DFHMAHRV copybook, which is in the SCIZMAC library.

Do not include any application data in this container.

Container DFHMAC-SYSPARMV1

The service requester uses DFHMAC-SYSPARMV1 to pass parameter values that override the defaults of the DFHMAH header and the contents of the DFHMAC-REQUESTV1 and DFHWS-DATA containers.

This container is mapped by copybook DFHMAHSV, which is in the SCIZMAC library. If a parameter value is non-blank, that value overrides the default.

Field	Length (bytes)	Type	Value
DFHMAH-PROCESSTYPE	8	Character	PROCESSTYPE resource
DFHMAH-PROCESSNAME	36	Character	Process name
DFHMAH-REQUESTNAME	8	Character	Request name of the service flow

Container DFHMAC-USERDATA

The service requester uses DFHMAC-USERDATA to pass application data to the CICS SFR interface program, DFHMADPL.

The service requester can send this optional container with the DFHMAC-REQUESTV1 or DFHMAC-ALLPARMS containers. When the service flow successfully processes the request message, any application data that is required in the response is returned in this container.

The length of the container can be up to, but not equal to, 16 MB.

Container DFHWS-DATA

DFHWS-DATA passes the request name and application data to the CICS SFR interface program, DFHMADPL, when a Web service request is received and processed in a provider mode pipeline.

The request name is eight characters in length. By default, the request name is the same as the PROCESSTYPE resource for the service flow. When the pipeline application handler passes this container to DFHMADPL, request processing uses the default values for the header. You can override these default values by passing in additional containers.

The length of this container can be up to, but not equal to, 16 MB.

Request message headers

The request message headers, along with the application data, form the request message that the service requester sends in a COMMAREA.

The following headers are described:

- DFHMAH message header
- DFHMAH2 passthrough message header
- CIA-SCREEN-HEADER passthrough screen header
- CIA-MAP-HEADER passthrough map header

DFHMAH header structure

The request message header DFHMAH describes the structure of the application data. It is mapped by copybook DFHMAHV, which is in the library SCIZMAC.

The following table highlights field information in the message header structure DFHMAH.

A field's type can be one of the following:

X
The field contains characters.

9
The field is a numeric

FB
The field is a fullword binary.

Table 3. DFHMAH message header fields

Disp.	Length	Type	Field	Req.	Values
0	04	x	DFHMAH-STRUCID	Y	MAH '<?XM', '<?xm' '<SOA', '<soa'
4	04	FB	DFHMAH-VERSION	Y	1
8	04	FB	DFHMAH-STRUCLENGTH	Y	384
12	08	x	DFHMAH-USERID	N	
20	08	x	DFHMAH-FORMAT	N	Spaces
28	04	FB	DFHMAH-RETURNCODE	N	
32	04	FB	DFHMAH-COMPCODE	N	
36	04	FB	DFHMAH-MODE	N	
40	04	FB	DFHMAH-SUSPSTATUS	N	
44	04	x	DFHMAH-ABENDCODE	N	
48	08	x	DFHMAH-MESSAGE	N	
56	04	x	DFHMAH-MSG-RESERVED	N	
60	04	FB	DFHMAH-UOWCONTROL	Y	zero, 1, 2, 9
64	08	x	DFHMAH-PROCESSTYPE	Y	

Table 3. DFHMAH message header fields (continued)					
Disp.	**Length**	**Type**	**Field**	**Req.**	**Values**
72	36	x	DFHMAH-PROCESSNAME	N	
108	08	x	DFHMAH-REQUESTNAME	Y	
116	04	FB	DFHMAH-DATALENGTH	Y	
120	36	x	DFHMAH-FAILED-PROCNAME	N	
156	08	x	DFHMAH-FAILED-PROCTYPE	N	
164	04	x	DFHMAH-FAILED-TRANID	N	
168	48	x	DFHMAH-REPLYTOQ	N	
216	48	x	DFHMAH-REPLYTOQMGR	N	
264	24	x	DFHMAH-MSGID	N	
288	24	x	DFHMAH-CORRELID	N	
312	08	x	DFHMAH-FAILED-PROGRAM	N	
320	32	x	DFHMAH-FAILED-NODE	N	
352	04	FB	DFHMAH-LINKTYPE	N	Reserved.
356	04	FB	DFHMAH-MORE-DATA-IND	N	
360	04	FB	DFHMAH-BRIDGE-RC	N	
364	16	x	DFHMAH-STATETOKEN	N	
380	04	x	DFHMAH-RESERVED2	N	Reserved.

DFHMAH field definitions

The following definitions describe the content of each field in the DFHMAH message header.

DFHMAH-STRUCID
The structure identifier. This value indicates the structure of the message header. This field is always an input field and the initial value is MAH. Acceptable input for this field can be any of the following values:

- MAH, indicating that the message header is structured in flat file format
- <?XM or <?xm, indicating that the message header is structured in XML format
- <SOA or <soa, indicating that the message header is structured using Service Oriented Architecture (SOA) format

DFHMAH-VERSION
The structure version number. The value must be 2 for runtime processing. This field is always an input field and the initial value is 2.

DFHMAH-STRUCLENGTH
The length of the DFHMAH structure. The value must be 384. This field is always an input field and the initial value is 384.

DFHMAH-USERID
Reserved. This field is not implemented currently.

DFHMAH-FORMAT
The format name of the application data that follows the DFHMAH structure. The format name is also used for the reply message.

If the request message results in the generation of an error reply message, the error reply message has a format name of INCMPLTE. This field is an input field for requests and an output field for replies. The initial value of this field is blanks.

DFHMAH-RETURNCODE
The return code from CICS Service Flow Runtime processing. This return code describes the outcome of service flow request processing. The Compcode, Mode, Suspstatus, Abendcode, and Message fields can contain additional information. The value is one of the following:

0
Request processing completed normally.

9
An error occurred during request processing.

99
Multiple errors occurred during request processing.

999
An abend occurred during request processing.

Depending on the processing mode, a value other than zero in this field can indicate an incomplete BTS process. This process remains in an incomplete status until some action is taken. See the description of field DFHMAH-COMPCODE.

This field is an output field. The initial value of this field is zero.

DFHMAH-COMPCODE
The completion status of the BTS process instance. This field describes the outcome of the BTS process that implements an instance of the CICS Service Flow Runtime. This field is an output field and the initial value of this field is zero. See Developing with the BTS API for a description of the **CHECK ACQPROCESS** command.

DFHMAH-MODE
The processing state of the BTS process instance. This field describes the state (at the time that the reply was issued) of the BTS process that implements an instance of the CICS Service Flow Runtime. This is an output field and the initial value of this field is zero. See Developing with the BTS API for a description of the **CHECK ACQPROCESS** command.

DFHMAH-SUSPSTATUS
The suspend status of the BTS process. This field is an output field and the initial value of this field is zero. See Developing with the BTS API for a description of the **CHECK ACQPROCESS** command.

DFHMAH-ABENDCODE
This field contains the ABEND code if present. The value returned in this field depends on the DFHMAH-RETURNCODE field. This field is an output field and the initial value is blanks. See CICS messages for a complete list.

DFHMAH-MESSAGE
The error message returned from CICS Service Flow Runtime. The value returned in this field depends on the DFHMAH-RETURNCODE field. This field is an output field and the initial value is blanks. See “Messages and codes” on page 118 for a description of potential errors and for information on the user response to the error message.

DFHMAH-UOWCONTROL
This field indicates the processing mode. The value is one of the following:

0
Normal or default processing. This value is the initial value of the field. A value of 0 indicates standard request processing of a service flow, where a BTS process instance is created to run a request from the service requester.

2
If the BTS process fails, the process is canceled and a compensating flow runs instead.

9
If the BTS process fails, the process is canceled.

If the field contains a value of 2 or 9, the FAILED-PROCNAME and FAILED-PROCTYPE fields must be specified.

DFHMAH-PROCESSTYPE
This field defines the type of the new process instance. It is controlled by the service requester. DFHMADPL uses the value in this field when it starts the navigation manager with the **BTS DEFINE PROCESS** command.

This is an input field and the initial value is blanks. This field is used to determine the following:

- BTS repository and audit files used
- The audit level

Do not specify this field if the DFHMAH-UOWCONTROL field has a value of 9. Because of the relationship between the audit level and BTS process type, you might want to define multiple process types for each audit level or request type.

DFHMAH-PROCESSNAME
This field indicates the name of the new process instance. DFHMADPL uses the process name on the **BTS DEFINE PROCESS** command. This field ensures that each BTS process has a unique name. See Developing with the BTS API for a description. This field is an input or output field. The initial value of this field is blanks.

If no value is provided, the CICS Service Flow Runtime generates a unique identifier and returns it to the service requester. When generated, the user ID under which DFHMADPL is running, Eibtaskn, and the AbsTime values are concatenated in that order and are used as the process name.

Do not specify this field if the DFHMAH-UOWCONTROL field has a value of 9.

DFHMAH-REQUESTNAME
This field value is the name of the service flow that should be processed. The CICS SFR interface program DFHMADPL uses this value to read the service flow repository file to determine how to run the service flow. The value must correspond to the name of a deployed and enabled service flow.

The initial value of this field is blanks. Do not specify this field if the DFHMAH-UOWCONTROL field has a value of 9.

DFHMAH-DATALENGTH
The length of the inbound request or outbound reply application data following the DFHMAH header structure. This value is used to determine the size of the input data-container that contains the request application data.

It represents the fixed-format length of the inbound application request data or outbound application reply data, not including the length of any XML tags or XML declaration data if present. For XML request messages, this field indicates the length of the application request data when it is converted to COBOL.

DFHMAH-FAILED-PROCNAME
Failed process name. This field indicates the name of the failed process and is an input or output field. It is returned by CICS Service Flow Runtime when the value of the DFHMAH-RETURNCODE field is not zero. The service requester must specify this field when the DFHMAH-UOWCONTROL field indicates cancel or compensate. It is used to acquire a failed process on a BTS **ACQUIRE PROCESS** command.

DFHMAH-FAILED-PROCTYPE
Failed BTS process type. This field indicates the type of the failed CICS Service Flow Runtime process. This field is an input or output field. It is returned by the CICS Service Flow Runtime when RETURNCODE is not zero. The service requester must specify this field when the DFHMAH-UOWCONTROL field indicates cancel or compensate. It is used to acquire a failed process on a BTS **ACQUIRE PROCESS** command.

DFHMAH-FAILED-TRANID
Failed CICS Service Flow Runtime transaction. This field indicates the active transaction ID when the error occurred causing process failure. This field is an input or output field. It is returned by the CICS Service Flow Runtime when the value of field DFHMAH-RETURNCODE is not zero. The application can

use it in a compensating flow or custom error processing. The CICS Service Flow Runtime does not make use of this field other than to return a value to the service requester upon failure.

DFHMAH-REPLYTOQ
Name of the reply queue. This value is the name of the message queue to which the CICS Service Flow Runtime sends reply messages. This field is an input field. See IBM MQ applications reference for a description. The service requester must specify this field if the processing request mode is asynchronous and the WebSphere MQ-CICS bridge is being used to communicate between the CICS Service Flow Runtime and the service requester. Do not specify the ReplyToQ in the WebSphere MQ MQMD header structure.

This field can be loaded if you are implementing early reply processing with the queue name. The name is used for any early reply issued.

DFHMAH-REPLYTOQMGR
This is the name of the queue manager to which the CICS Service Flow Runtime sends reply messages. This field is an input field. *Replytoq* is the local name of a queue that is defined on this queue manager. See IBM MQ applications reference for a description. The service requester must specify this field when the processing mode is asynchronous and the WebSphere MQ-CICS bridge is being used to communicate between the CICS Service Flow Runtime and the service requestor. Do not specify the ReplyToQMgr in the WebSphere MQ MQMD header structure.

This field can be loaded if the service requester is implementing early reply processing with the queue manager name. The name will be used for any early reply issued.

DFHMAH-MSGID
This field can specify the unique message identifier (MSGID) used to put the CICS Service Flow Runtime request message on the WebSphere MQ-CICS bridge queue. During asynchronous processing, CICS Service Flow Runtime handles the reply message itself, rather than allowing the bridge to return the reply message. The bridge does not pass the WebSphere MQ headers to the CICS Service Flow Runtime. Therefore, when you want to correlate request and reply messages, you must explicitly supply a MSGID, rather than allowing it to be generated by the queue manager. When specified, the value is copied to the MQMD CorrelId on **MQPUT** commands issued from server adapters and on any reply messages to the service requester. This field is an input field.

DFHMAH-CORRELID
This field can specify the correlation identifier used to put the CICS Service Flow Runtime request message on the WebSphere MQ-CICS bridge queue. The bridge does not pass the WebSphere MQ headers containing the CORRELID to the CICS Service Flow Runtime. Therefore, when you want your adapters to know the correlid with which the message was put on the queue, you must explicitly supply a CORRELID, rather than allowing it to be generated by the queue manager. This field is an input field. The WebSphere MQ-CICS bridge requires a Correlid of MQCI-NEW-SESSION on the first message of any unit of work.

DFHMAH-FAILED-PROGRAM
Failed program name. This field indicates the name of the active program when an error occurred causing process failure. This field is an input or output field. It is returned when the value of field DFHMAH-RETURNCODE is not zero. The application can use it in a compensating flow or custom error processing. The CICS Service Flow Runtime does not make use of this field other than to return a value to the service requester upon failure.

DFHMAH-FAILED-NODE
This field indicates the name of the active node when an error occurred causing process failure. This field is an input or output field. It is returned by the CICS Service Flow Runtime when the value of field DFHMAH-RETURNCODE is not zero. The application can use it in a compensating flow or in custom error processing. The CICS Service Flow Runtime does not make use of this field other than to return a value to the service requester upon failure. See the information on the service flow project tools in the IBM Developer for Z information center for detailed information on defining nodes.

DFHMAH-LINKTYPE
Reserved. Do not set a value for this field.

DFHMAH-MORE-DATA-IND
This field indicates that additional response data is available but not deliverable. It is returned by the CICS Service Flow Runtime when the response data is larger than 32 000 bytes. This field is an output field. The value is one of the following:

- N = no additional response data
- Y = additional response data

DFHMAH-BRIDGE-RC
This return code is returned by the Link3270 bridge to the CICS Service Flow Runtime. It is returned by the CICS Service Flow Runtime when the value of field DFHMAH-RETURNCODE is not zero. This field is an output field. The initial value of this field is zero.

DFHMAH-STATETOKEN
This field value, if present in a CICS Service Flow Runtime reply message, indicates that Link3270 server adapter processing has left an allocated Link3270 bridge facility with associated facility business state data stored for subsequent reuse. This field is an input and output field. It is returned when request processing is complete. It can be passed in subsequent input request messages to retrieve facility business state data for reuse in Link3270 server adapter processing. The initial value of this field is blanks.

See Administering the Link3270 bridge for more information on Link3270 bridge facilities.

DFHMAH-RESERVED2
Reserved. Do not set a value for this field.

Chapter 7. Managing service flows

When you have successfully deployed one or more service flows into CICS, you can use the flow management transaction to administer them.

About this task

The flow management transaction enables you to browse the contents of the service flow repository file DFHMAASF, which contains a record for every service flow that has been deployed into the runtime environment.

CMAN - the flow management transaction

Use the CMAN transaction to manage your deployed service flows. You can install new service flows, enable and disable installed service flows, and delete service flows from the service flow repository file. You can also use this transaction to view and change the options for a particular service flow, such as turning vector logging on and off.

To start the CMAN transaction, you can do one of the following:

- Enter CMAN on the command line of your display. The transaction lists, in alphabetical order, all the deployed service flows that are defined in the DFHMAASF service flow repository file. CMAN also displays the status of each service flow and the number of times it has been invoked since the CICS region started up.
- Enter CMAN on the command line of your display, followed by the required action and service flow:

```
CMAN DISABLE FLOW(INQUIRE1)
```

If you enter CMAN on the command line of your display, the main screen displays:

```
DFHMA00       CICS TS V3.2 Service Flow Runtime - Installed Service Flows
Request name filter: *                                                More: + -
Actions: 1= View details 2= Enable 3= Disable 4= Delete

Act   Request name   Status   Use count   Response
 _      INQUIRE1     Enabled   00006
 _      ITEMREQ      Enabled   00000
 _      NACTTEST     Disabled  00058
 _      NCFNAVS      Enabled   00377
 _      NCLNAVA      Unusable  00766
 _      NCMQAA       Enabled   00111
 _      ORDCREQ      Enabled   00118
 _      REQFEPI      Enabled   00118


DFHMA100I  Type action code then press ENTER or F2 to install flows
F1=Help  F2=Install Flows  F3=Return  F7=Page Up  F8=Page Down  F12=Cancel
```

Figure 4. CMAN transaction: main screen

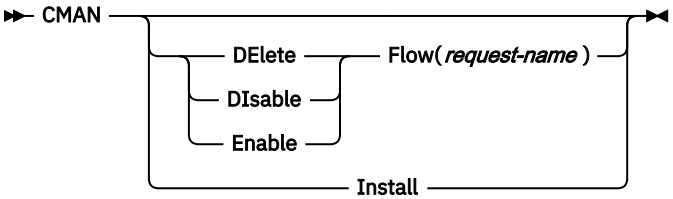
From this screen you can perform a number of tasks:

- Scroll through the list of installed service flows using the PF7 and PF8 keys.
- View a subset of service flows, using the Request name filter field and the wildcard character *.
- View the details of a service flow, including the list of its server adapters.
- Enable, disable, and delete one or more service flows.

If you require help, press PF1.

Command syntax

The command syntax of the CMAN transaction is as follows:



DElete
Delete a service flow from the service flow repository file. You can delete a service flow only when its status is disabled. The service flow properties file is not deleted from the zFS deployment directory and the associated CICS resources are not deleted.

DIsable
Disable a service flow. You can disable a service flow only when its status is enabled. The CMAN transaction disables the PROCESSTYPE resource that is associated with the service flow so that the service requester cannot invoke it.

Enable
Enable a service flow. You can enable a service flow when its status is disabled. The CMAN transaction enables the PROCESSTYPE resource that is associated with the service flow so that it can be invoked by a service requester.

Install
Install any new service flow properties files. When you perform this action, the CMAN transaction scans the zFS deployment directory for new service flows and defines them in the service flow repository file.

Flow(*request-name*)
The *request-name* is the name of the service flow against which you want to perform the action.

Viewing the details of service flows

Use the CMAN transaction to view the server adapters that are part of a service flow and the properties associated with each server adapter.

Procedure

1. Enter CMAN to display the list of service flows that are installed.
2. Enter action 1 next to the service flow that you want to view.
   The following screen is displayed:

```
DFHMA10       CICS TS V3.2 Service Flow Runtime - ServiceFlow Details
                                                              More: +-
Request name: INQUIRE1
Vector logging actions(Link3270 nodes): 1= Full 2= Trace 3= Off

        Server     Adapter
Act     adapter      type          Attributes  Response
 _      Link1      Subflow
 _      Prog2      Subflow
 _      Screen1    FEPI            Exit=Hold
 _      Screen2    FEPI            Exit=Assigned
 _      WebS1      FEPI            Exit=Release
 _      Link2      Link3270        Vlog=Off



F1=Help   F3=Return   F7=Page Up  F8=Page Down   F12=Cancel
```

For each server adapter that is used in the service flow, the name of the program and the type of adapter is displayed.

- For Link3270 server adapters only, this screen displays the status of the vector logging.
- For FEPI server adapters, this screen displays the exit processing.

If a server adapter is listed as a subflow, it indicates that the service flow calls another service flow as part of its request processing.

Updating an existing service flow

If you need to update an existing service flow, you should perform the following steps.

Procedure

1. Use the CMAN transaction to disable the service flow.
2. If you are changing the resources associated with the service flow, for example a different transaction or program, delete the resources from the CSD.
3. Regenerate the service flow and run through the deployment steps as outlined in "Deploying a new service flow" on page 33.
4. If you have recompiled any programs, use the **CEMT SET PROGRAM** with the NEWCOPY attribute.
   This loads the latest version of the program into memory for all new transactions requests.
5. Use the CMAN transaction to enable the service flow.

Disabling access to service flows

Use the CMAN transaction to disable access to deployed service flows.

About this task

You can use either of the following methods to disable a service flow:

Procedure

- To disable a service flow, enter `CMAN DISABLE FLOW(`*`request-name`*`)`, where *request-name* is the name of the service flow.
- To disable one or more service flows, enter `CMAN` to display the list of installed service flows.
  a) Enter 3 in the Action column next to the service flows that you want to disable.
     To disable a service flow, it must be in the enabled status.
  b) Press the Enter key.
     The DFHMAINS program disables the PROCESSTYPE resource that is associated with the service flow. A message is issued to the console.
  c) Press the Enter key to refresh the list of service flows.
     The service flows that you selected are listed as disabled.

Results

Any service requesters that try to invoke a disabled service flow receive a DFHMA prefixed error message.

What to do next

If you want to enable a service flow that is disabled, use the CMAN transaction and option 2.

Deleting service flows

Use the CMAN transaction to delete one or more service flows from the service flow repository file DFHMAASF.

Before you begin

You can delete only the service flows that are in a disabled state.

About this task

You can delete a service flow using either of the following methods:

Procedure

- To delete a particular service flow, enter `CMAN DELETE FLOW(`*`request-name`*`)` on your display, where *request-name* is the name of the service flow.

  The transaction runs the DFHMAINS program to delete the service flow from the service flow repository file.
- To delete more than one service flow, enter `CMAN` on your display.

  The transaction displays the flow management main menu where you can perform actions on service flows.

  a) Enter option 4 next to each disabled service flow that you want to delete.

  b) Press Enter.

  The DFHMAINS program checks that each service flow is disabled and then deletes it from the service flow repository file. A message on the console confirms that the deletion was successful.

  c) Press Enter to refresh the view.

Results

The DHFMAINS program does not delete subflows that are associated with a service flow when it is deleted from the repository file. The resource definitions, programs, and service flow properties file are also not deleted.

What to do next

To remove a service flow from the CICS region completely, you must delete all of the CICS resources for the service flow from the CSD and remove the service flow properties file from the service flow deployment directory in zFS.

Service flow recovery on a CICS restart

If you restart the CICS Service Flow Runtime region, the DFHMAINS PLT program can ensure that any deployed service flows are available at the end of the system initialization process.

If you have specified `'NO_SFR_INST'` on the **`INITPARM`** parameter for the region, no service flows install when you restart CICS. If you do specify a deployment directory on the **`INITPARM`** parameter, the processing that takes place depends on how the CICS region starts up.

Cold or initial start

If you perform a cold or initial start of the CICS region, the DFHMAINS program acts as follows:

1. Deletes all of the records in the service flow repository file DFHMAASF.
2. Reads the service flow files in the zFS deployment directory and defines them in DFHMAASF. You specify this directory on the INITPARM statement for the DFHMAINS program. The service flow files are identified by the file extension `.sfp`.
3. Installs the service flow files that are contained in the zFS deployment directory.

4. Creates the resource definitions for the service flows. This resource definition occurs only if selected during the deployment of the service flow in IBM Developer for Z.

When DFHMAINS finishes, the message DFHMA00002I is issued to the CICS console and log.

Warm start

If you perform a warm start of the CICS region, the DFHMAINS program acts as follows:

1. Checks the zFS deployment directory for any new or updated service flow files.
2. Installs the new or updated service flow files in CICS and updates the service flow repository file DFHMAASF.
3. Creates the resource definitions required by the service flows. This resource definition occurs only if selected during the deployment of the service flow in IBM Developer for Z.

When DFHMAINS finishes, the message DFHMA00002I is issued to the CICS console and log.

Emergency start

If you cancel the CICS region to perform an emergency restart or if the CICS region abends, the DFHMAINS program does not check the zFS deployment directory for new or updated service flow files. All previously installed service flows are made available, but new service flows are not installed.

To install new service flows and optionally create the resource definitions, use the flow management transaction CMAN.

Server runtime utilities

The table lists sample jobs and utilities that can help you administer and maintain the CICS Service Flow Runtime.

Most of these utilities are provided as samples. They are in the samples library *hlqual*.SCIZSAMP, which was created when you ran the setup procedure DFHMAINJ. You must compile the sample programs that are provided before they can be executed. See “Performing postinstallation tasks” on page 21 for further information.

Table 4. CICS Service Flow Runtime utilities

Name	Description
DFHMAINA	Sample program that applies APARs to the runtime environment.
DFHMAMVD	Sample JCL to run the Link3270 vector log file dump DFHMAVUP See “Link3270 Vector Log file dump JCL, DFHMAMVD” on page 154 for further information.
DFHMAVUP	Sample program that dumps the active Link3270 vector log file, either DFHMALVA or DFHMALVB. See “Vector file dump” on page 154 for further information.

Chapter 8. Processing service flows

The CICS Service Flow Runtime processes service flow requests synchronously and asynchronously and can support five different processing modes. Each service flow that is generated by service flow project tools defines which processing mode is required, although the service requester can override the mode in the request message.

A service requester can use any of the supported interfaces to invoke a service flow synchronously and can invoke a service flow asynchronously using WebSphere MQ and the WebSphere MQ-CICS bridge. CICS SFR can also process a service flow request asynchronously, even if the service requester invoked the service flow synchronously.

The processing of a service flow, or *request processing*, conforms to a processing pattern. Even though simple and complex service flows can contain many server adapters of different types, the methods for processing them are the same.

Processing modes

There are four different ways that a service flow request can be processed by CICS SFR. The processing mode of the service flow is set in service flow project tools, but can be overridden by the service requester when it invokes the service flow.

Each mode is a combination of the processing required by the service requester and the processing of the BTS activities that run the flow navigator and the server adapters.

Table 5. Processing modes

Mode name	Mode number	Service requester mode	Activity mode
Asynchronous	0	Asynchronous	Asynchronous
Link	4	Synchronous	Synchronous using LINK ACTIVITY
Synchronous	1	Synchronous	Synchronous
Synchronous roll back	2	Synchronous	Synchronous with additional error processing

By default, service flow project tools processes requests in synchronous mode. The processing associated with each mode is as follows:

Asynchronous mode
The service requester must be a WebSphere MQ-enabled application that uses the WebSphere MQ-CICS bridge to invoke a service flow asynchronously. The CICS SFR interface program, DFHMADPL, initiates the BTS process to start the navigation manager asynchronously and returns to the bridge task. The bridge task also returns, committing the **GET** of the request message.

The navigation manager invokes the flow navigator and waits for it to complete its processing. The flow navigator runs all of the server adapters asynchronously and handles the completion events of the server adapters until all of the request processing is complete. The flow navigator returns to the navigation manager, which in turn reads any reply message from the flow navigator's output data-container, and puts the reply message on the queue specified in the REPLYTOQ field of the request message.

Link mode
The service requester invokes a service flow synchronously and DFHMADPL defines the BTS process for the navigation manager synchronously. The request processing is also synchronous, but the

navigation manager and flow navigator use an **EXEC CICS LINK ACTIVITY** command for creating child activities.

Use Link mode if you have a large service flow with many server adapters. All of the activities run in the same unit of work, so there are fewer tasks for the service flow, but an error in any server adapter causes all of the updates to roll back to the state prior to the start of the failed request.

Synchronous mode
The service requester invokes a service flow synchronously and DFHMADPL defines the BTS process for the navigation manager synchronously. The navigation manager waits for the flow navigator and server adapter processing to complete before returning to DFHMADPL. DFHMADPL sends a response message to the service requester. When the BTS process completes, the unit of work is committed, even if there are system errors.

Synchronous roll back mode
This processing mode is the same as the synchronous mode with one exception. If a failure occurs in any activity within the BTS process, the unit of work is not committed. If an error occurs, all recoverable resources that are updated during the request processing are returned to the state prior to the start of the failed request. For example, if your service flow has a series of DPL requests that run as a single unit of work and are committed and if any one of the DPL requests is unsuccessful, the unit of work is not committed.

Processing patterns

Processing patterns support the deployment patterns that are used by service flow project tools to generate the service flows.

Request processing patterns for simple service flows

Simple service flows conform to the single connector pattern. Simple service flows are processed in slightly different ways, depending on whether the service flow is persistent. If the service flow is not persistent, the command that DFHMADPL uses to create the navigation manager process has an additional option.

The following steps describe the request processing of a simple service flow:

1. A service requester invokes the service flow when it sends a request message using any of the supported interfaces.
2. The CICS SFR interface program DFHMADPL receives the request and, using the request name in the request message, DFHMADPL reads the service flow repository file DFHMAASF.

   If the request is in XML, DFHMADPL invokes the XML-to-COBOL converter program to parse the XML header and return COBOL. DFHMADPL then reads the service flow repository file.
3. The process mode of the service flow is defined in the service flow repository file. If the service flow is simple, DFHMADPL performs the following processing:
   a. Defines the BTS process using the BTS command **EXEC CICS DEFINE PROCESS**. If the service flow is persistent, a record is added to the BTS repository for the process, allowing CICS to recover if a failure occurs. If the service flow is not persistent and the BTS NOCHECK option is specified, a record is not added to the BTS repository.
   b. Writes application request data to the ADAPTER.INPUT data-container. The server adapter reads the ADAPTER.INPUT data-container to retrieve the application request data that is used to process the service flow.
   c. Writes state information to the ADAPTER.PROCESS data-container. State information includes the request name, the program name, and the transaction that will process the application request data. For service flows of the single connector type, the program and transaction is a FEPI or Link3270 server adapter.
   d. Runs the BTS process in synchronous or asynchronous mode, as defined in the service flow repository file.
4. After the navigation manager is invoked, it performs the following processing steps:

a. Reads the state information from the ADAPTER.PROCESS data-container.

b. Defines the BTS activity that processes the request to run the service flow, using the BTS command **EXEC CICS DEFINE ACTIVITY**:

c. Writes adapter state information to the ADAPTER.PROCESS data-container. Additional state information includes the activity ID, the CICS applid where the Navigation Manager is running, and the EIBTASKN of the running Navigation Manager.

d. Runs the BTS activity for the service flow in synchronous or asynchronous mode, as defined in the service flow repository file.

When the Navigation Manager runs the BTS activity, the FEPI or Link3270 server adapter performs request processing to complete the request made by the service requester. Each business request issued by the service requester results in a different BTS process instance to run the appropriate service flow. Each process instance consists of a Navigation Manager activity and the server adapter activity that is needed to support that service flow.

Reply processing pattern for a simple service flow

When a Link3270 or FEPI server adapter completes its processing, the application response is stored in a data-container called ADAPTER.OUTPUT. The status of the server adapter processing is also stored in a data-container called ADAPTER.STATUS.

After the server adapter has completed processing and the service flow activity has ended normally, a BTS completion event returns control to the navigation manager, DFHMAMGR. The navigation manager performs the following processing steps:

1. Performs a check on the server adapter activity completion status by issuing the **CHECK ACTIVITY** command:

```
EXEC CICS CHECK ACTIVITY (ADC-NAV-ACTIVITY)
               COMPSTATUS (CICS-COMPSTATUS)
                   ABCODE (CICS-ABCODE)
                ABPROGRAM (CICS-ABPROGRAM)
                     MODE (CICS-MODE)
               SUSPSTATUS (CICS-SUSPSTATUS)
                     RESP (CICS-RESP)
                    RESP2 (CICS-RESP2)
END-EXEC.
```

2. Reads the appropriate server adapter status information from data-container ADAPTER.STATUS. If the server adapter completed successfully, the navigation manager reads the output container ADAPTER.OUTPUT.
3. Writes the root activity output data-container containing the application response or reply data. The output container name is ADAPTER.OUTPUT.
4. Writes adapter completion status and state information to process container ADAPTER.PROCESS.

After the Navigation Manager has completed processing and the BTS process has ended normally, a BTS completion event returns control to DFHMADPL.

DFHMADPL performs the following processing steps:

1. Performs a check on the process completion status by issuing the **CHECK ACQPROCESS** command.
2. Reads the adapter state information from process container ADAPTER.PROCESS
3. Reads the output data-container ADAPTER.OUTPUT.
4. Builds a reply message that includes any output application data and sends it to the service requester.

   DFHMADPL can invoke a user-defined program to convert the application data to XML. The used-defined program name is determined by the service flow. DFHMADPL then calls the COBOL to XML converter program, DFHMAXMO, to convert the COBOL header structure (DFHMAH) to XML, to format the XML reply message.

Request processing patterns for complex service flows

Complex service flows conform to the aggregate connector pattern. Complex service flows are processed in slightly different ways, depending on whether the service flow is persistent. If the service flow is not persistent, the command that DFHMADPL uses to create the navigation manager process has an additional option.

The following steps describe the request processing of a complex service flow:

1. A service requester invokes the service flow when it sends a request message using any of the supported interfaces.
2. The CICS SFR interface program DFHMADPL receives the request and, using the request name in the request message, reads the service flow repository file DFHMAASF.

   If the request is in XML, DFHMADPL invokes the XML to COBOL converter program to parse the XML header and return COBOL. DFHMADPL then reads the service flow repository file.
3. The processing mode of the service flow is defined in the service flow repository file. If the service flow is complex, DFHMADPL performs the following processing:
   a. Defines the BTS process using the BTS command **EXEC CICS DEFINE PROCESS**. If your service flow is persistent, a record is added to the BTS repository for the process, allowing CICS to recover if a failure occurs. If the service flow is not persistent and the BTS NOCHECK option is specified, a record is not added to the BTS repository.
   b. Writes application request data to the process input data-container ADAPTER.INPUT. The flow navigator program reads the input data-container ADAPTER.INPUT to retrieve the application request data that is used to run the service flow.
   c. Writes state information to the ADAPTER.PROCESS data-container. State information includes the request name, the program name, and the transaction that will process the application request data. For aggregate service flows, the program and transaction is the flow navigator. The flow navigator manages the order and processing of each server adapter, as modeled in the flow.
   d. Runs the BTS process in synchronous or asynchronous mode, as defined in the service flow properties file, to start the navigation manager.
4. The navigation manager performs the following processing:
   a. Reads the adapter state information from the ADAPTER.PROCESS data-container.
   b. Defines the flow navigator BTS activity that will process the application request, using the **EXEC CICS DEFINE ACTIVITY** command:
   c. Writes adapter state information to the ADAPTER.PROCESS data-container. Additional state information includes the activity ID, the CICS applid where the navigation manager is running, and the EIBTASKN of the navigation manager.
   d. Runs the flow navigator as defined by the service flow, either in synchronous or asynchronous mode.

When the Navigation Manager runs the flow navigator, the server adapters perform request processing as modeled in service flow project tools, to complete the request made by the service requester.

For example, a deployed service flow, with 1 to *n* server adapters of any type, might perform the following steps:

1. Invoke a server adapter to interact with a CICS and IMS application using a 3270 data stream and retain the result. This processing consists of screen navigation.
2. Invoke a server adapter to interact with a WebSphere MQ-enabled application and post the result of the first server adapter's processing to a WebSphere MQ queue.
3. Invoke a server adapter to interact with one or more CICS transactions using DPL and send the result of the first server adapter. For example, it could write a record to a DB2® database.
4. Invoke a custom program using DPL that executes logic or complex input and output. You can develop a custom program to augment CICS Service Flow Runtime.

5. Return application data to the navigation manager. If this data contains the final result, the navigation manager returns to the CICS SFR interface program to pass the response back to the service requester.

Each business request issued by the service requester results in a different BTS process instance to run the appropriate service flow as defined by the service requester. Each process instance consists of a navigation manager activity, a flow navigator activity, and server adapter activities that support the invoked service flow.

Reply processing pattern for complex service flows

When each server adapter completes normally, it stores the application response and its status in data-containers and returns control to the flow navigator.

Program link and Queue server adapters put status information in a COMMAND.STATUS data-container. FEPI and Link3270 server adapters put status information in a NAVIGATOR.STATUS data-container.

The flow navigator performs the following processing steps each time a server adapter completes:

1. Performs a check on the server adapter activity completion status by issuing the **BTS CHECK ACTIVITY** command.
2. Reads the adapter state information from the ADAPTER.PROCESS data-container.
3. Reads the server adapter status information from the appropriate data-container.
4. Reads the appropriate server adapter output data-container, if it successfully completed.

If the service flow has completed processing successfully, the flow navigator performs the following reply processing steps:

1. Puts the application response data in the ADAPTER.OUTPUT data-container.
2. Updates the data-container ADAPTER.STATUS to indicate that the service flow completed successfully.
3. Issues an **EXEC CICS RETURN ENDACTIVITY** command, returning control to the navigation manager.

The navigation manager performs the following reply processing:

1. Checks the flow navigator completion status by issuing the **CHECK ACTIVITY** command.
2. Reads the state from the ADAPTER.PROCESS data-container.
3. Reads the service flow status from the ADAPTER.STATUS data-container.
4. Reads the service flow container from the ADAPTER.OUTPUT data-container, if the flow navigator completed successfully.
5. Puts the application response in the root activity data-container. This data-container is also called ADAPTER.OUTPUT.
6. Writes adapter completion status and state information to data-container ADAPTER.PROCESS.
7. Optionally issues an **MQPUT** command if the client application interface is WebSphere MQ-CICS bridge. Otherwise, the navigation manager issues an **EXEC CICS RETURN ENDACTIVITY** command, returning control to the CICS SFR interface program DFHMADPL and completing the process.

The CICS SFR interface program performs the following processing steps:

1. Performs a check on the process completion status by issuing the **CHECK ACQPROCESS** command.
2. Reads the state information from data-container ADAPTER.PROCESS.
3. Reads the navigation manager output data-container ADAPTER.OUTPUT.
4. Builds a reply message, including the application response data.
5. Moves the outbound reply message to a COMMAREA or container and issues an **EXEC CICS RETURN** command, returning control to the service requester.

Server runtime processing and the BTS NOCHECK option

The BTS NOCHECK option specifies that no record will be written to the BTS repository data set to reserve the name of the process. Using the BTS NOCHECK option improves BTS performance by removing the need to update the repository and its associated logging.

You must know the following information regarding the BTS NOCHECK option because it is relevant to CICS Service Flow Runtime processing.

- The BTS NOCHECK option is enabled only if the persistence is set appropriately. If the service flow is persistent, the NOCHECK option is omitted. If the service flow is not persistent, the NOCHECK option is used.

  This option is defined in the service flow when it is modeled and included in the service flow properties file. The CICS SFR interface program DFHMADPL uses this information when defining the BTS process to run the service flow.
- If you use the BTS NOCHECK option, be aware that the error of specifying a non-unique process name might not be discovered until a syncpoint occurs, making it much harder to debug.

Program link server adapter processing

When the program link server adapter DFHMASDP is invoked during request processing, the flow navigator passes the details of the distributed programming link to the server adapter in BTS data-containers.

DFHMASDP runs by default under the CMAS transaction and performs a program link to a target application using the **EXEC CICS LINK** command. DFHMASDP can pass the data in a COMMAREA that can be up to 32 767 bytes in length or by using a channel and containers. The target application might be in the same region or in a remote region.

The flow navigator passes a number of data-containers to DFHMASDP that store the data that is specified by service flow project tools when the link is modeled. The number of data-containers depends on whether the program link has been modeled to use a COMMAREA or a channel.

Program links that use a COMMAREA
: The flow navigator puts the data to be passed in the COMMAREA in the COMMAND.INPUT data-container. The flow navigator puts the description of the program link itself in the DPL.DATA data-container. DFHMASDP uses the data in COMMAND.INPUT to populate the COMMAREA and issues an **EXEC CICS LINK** command to the target application.

  When DFHMASDP receives a response from the application in a COMMAREA, it stores the data in the COMMAND.OUTPUT data-container and the state information in the COMMAND.STATUS data-container.

Program links that use a channel
: The flow navigator creates BTS data-containers of the same name and with the same content as every container that is required on the channel. The flow navigator puts the name of the channel, the name of each container, and the number of containers in the DPL.DATA data-container. The default channel name is DFHMA-DPL-CHNL. You can override the default channel name in service flow project tools. DFHMASDP uses the information in DPL.DATA to create the channel and the information from the BTS data-containers to create each container and put it on the channel.

  When DFHMASDP receives a response from the application on the channel, it updates the COMMAND.OUTPUT and COMMAND.STATUS data-containers. In addition, DFHMASDP copies each container on the channel to a BTS data-container of the same name. If any required containers are missing or if there is a problem copying the container, DFHMASDP issues an error message and returns to the flow navigator.

If the target application runs in a different region from the CICS Service Flow Runtime, the target application program runs under CSMI, the default mirror transaction ID. You can optionally specify a different transaction in service flow project tools. If you do specify a different transaction, ensure that it is defined in CICS to invoke the DFH$MIRS mirror transaction program.

If the target application includes any DB2 calls, you must configure the transaction under which the target application runs, whether this is CMAS, CSMI, or your own transaction, as follows:

1. Create the necessary DB2TRANS or DB2ENTRY resource definitions for the transaction.
2. Grant access authority in DB2 and RACF as appropriate.

Using DPL in a FEPI or Link3270 server adapter

If the FEPI or Link3270 bridge server adapter uses an inline DPL command, the runtime behavior for the DPL command differs from that of a DPL command used in a regular adapter.

When the FEPI server adapter or Link3270 bridge server adapter is generated, the adapter contains an embedded **EXEC CICS LINK** command for the DPL command, instead of the generated statements that define and perform a new BTS activity to execute the link.

In Figure 5 on page 65, *linkname* contains the linkname from the MAT_LINKNAME property in the properties file. The *command input message* variable is the DPL Command node input terminal message. The *sysid* variable is the MAT_SYSID property from the Connection Resource file.

```
EXEC CICS LINK PROGRAM (linkname)
      COMMAREA (command input message)
        LENGTH  (length of command input message)
        SYSID   (sysid)
        RESP    (CICS-RESP)
        RESP2   (CICS-RESP2)
      END-EXEC.

      IF CICS-RESP NOT EQUAL 0
        MOVE +9                 TO ERROR-IND
        MOVE DPL-ERRMSG         TO WS-ERR-MESSAGE
        MOVE DPL-ERROR-CODE     TO WS-ERR-CODE
        MOVE CICS-RESP          TO EDC-DPL-RESP
        MOVE CICS-RESP2         TO EDC-DPL-RESP2
        MOVE linkname           TO EDC-DPL-PROGRAM
        MOVE sysid              TO EDC-DPL-SYSID
        MOVE SPACES             TO EDC-DPL-TRANSID
        MOVE SPACES             TO EDC-DPL-SYNCONRETURN
        MOVE LENGTH OF command input message
                                TO EDC-DPL-LENGTH
        MOVE LENGTH OF command input message
                                TO EDC-DPL-DATALENGTH
        MOVE command input message
                                TO EDC-DPL-DATA
        PERFORM POST-NAVIGATOR-ERROR-RTN
          THRU POST-NAVIGATOR-ERROR-EXIT
        PERFORM 9010-NAVIGATOR-RETURN.
```

Figure 5. Invocation of DPL in FEPI service flow

The **EXEC CICS LINK** for the inline DPL command runs under the same activity as the FEPI or Link3270 server adapter itself. Therefore, the FEPI or Link3270 service adapter issues a direct link to the user-written program as part of the current activity (unit-of-work boundary) as opposed to defining and starting a new BTS activity as occurs for DPL commands in a regular (that is, non-FEPI or non-Link3270) service flow.

Failure of an inline **EXEC CICS LINK** to complete successfully (for example, PGMIDERR) is handled in the same manner as a LINK failure in a generated DPL command program. From a modeling perspective, runtime error handling for a DPL Command node in a FEPI or Link3270 service flow is the same as a DPL Command node in a regular service flow.

The **SYNCPOINT** and **SYNCPOINT ROLLBACK** commands cannot be executed in the linked-to program or any of its subprograms. Also, using the **ABEND** command in the linked-to program or any of its subprograms might cause an undesirable state of the current activity.

FEPI server adapter processing

FEPI server adapters can be part of simple or complex service flows and can therefore be invoked by the navigation manager directly or by a flow navigator. When the FEPI server adapter is invoked during

request processing, the 3270 data stream is passed in BTS data-containers to the FEPI server adapter for processing.

The FEPI server adapter uses the information in the BTS data-containers to start a session using the FEPI pool and node as modeled in service flow project tools. The server adapter handles the outbound screen from the FEPI application to the target application, identifying the fields, attributes and data, and the response from the target application. Depending on what is modeled, the adapter can handle multiple screens. It also manages state information about the status of the logical units.

You can model flows running on any 3278 model up to model 5 (27 x 132 screen size). You must set up a FEPI property set with a device type for each 3270 model terminal that you want to use. Use the DATASTREAM option, because CICS Service Flow Runtime uses that to convert from the terminal data to the COBOL representation of the screen. Also specify the MAXFLENGTH in the property set. The MAXFLENGTH must be large enough to contain the largest send or receive 3270 datastream for the transactions you are running in the flow. If the MAXFLENGTH is too small, CICS issues message DFHMA04012E and the flow fails.

FEPI server adapters can use any pool size that has a MAXFLENGTH value of up to and including 25 000 bytes. However, the MAXFLENGTH value helps FEPI use storage more efficiently and you must therefore set this value no larger than is necessary. The property set for the FEPI server adapters should not contain the FEPI sample transactions, as these are not supported in CICS Service Flow Runtime. For example, CZUC, CZUU, CSZX, and CZUX must not be included in the property set.

A FEPI server adapter uses several files to record information about its processing and to pass information to subsequent FEPI server adapters. These are as follows:

Target interaction file
: The target interaction file is a VSAM data set called DFHMATIF. It stores the COBOL representation of the last screen that was received in a FEPI adapter and some additional information in a buffer. The COBOL representation can be up to 3564 bytes. The additional information includes the cursor position, last attention key, current screen size, and the MAXFLENGTH value. Another FEPI server adapter uses this buffer in the service flow. If the first FEPI server adapter does not sign out of the terminal, the next FEPI adapter can acquire ownership of the conversation and uses the buffer to determine the state of that conversation.

  This file is used only in complex service flows. It can contain a buffer of up to 25000 bytes.

Connection file
: The connection file is a VSAM data set called DFHMACOF that contains the state of connections and conversations, and the ownership of those conversations, for a FEPI server adapter. It identifies whether a terminal (FEPI node) is in use and if that terminal can be used in subsequent FEPI server adapters.

  This file has an associated alternate index. The file is called DFHMAC1F.

You can model a service flow with many FEPI subflows, where each subflow generates a FEPI server adapter. The options that are set in each subflow for logging off the terminal affect the FEPI server adapter processing and how it passes information to the next server adapter:

- If the logoff option is `force`, `hold`, or `release`, the defined target and node that is in use by a FEPI server adapter is deleted when the server adapter has finished its processing. No buffer is written to the target interaction file.
- If the logoff option is `pass`, the connection and conversation information in the connection file is rewritten to indicate that the existing conversation has no owner. The conversation is available for another FEPI server adapter to use. The last buffer that was sent or received is stored in the target interaction file.
- If the logoff option is `leave assigned`, the connection and conversation information is left assigned to indicate the owner, which is the user ID that is signed on to CICS and associated with the FEPI server adapter processing. The last buffer that was sent or received is stored in the target interaction file. This option allows the processing of FEPI server adapters in different tasks to run under the same user ID. When this option is specified, the flow navigator adds a unique tag to the alternate connection file DFHMAC1F.

Service flows with shared user IDs

If you have shared user IDs in your FEPI service flows, they affect the way the flow navigator uses logical units to establish connections and sessions to CICS regions and applications. You are recommended not to use shared user IDs in your service flows.

A shared user ID is not associated with a single user, but instead more than one user at a time can use it. Logical units (LUs) are required to establish connections and sessions to CICS regions and applications for FEPI server adapters. Assigning logical units to shared user IDs means the following:

- Logical units (LUs) in the same FEPI pool are assigned to a shared user ID for the duration of the service flow request. This might involve running many FEPI server adapters as part of the service flow.
- Many service flows might be running concurrently in the CICS region using the same shared user ID. Each invocation of a service flow runs in a separate BTS process.
- Logical units might remain signed on and in session for a shared user ID across many invocations of subsequent service flows that might have FEPI server adapters. The local unit can remain assigned to the shared user ID in the connection file after a service flow has completed.

In a configuration where the user ID is shared, CICS SFR ensures that logical units in the same FEPI pool are assigned uniquely to the shared user ID. A unique tag is added for each logical unit in the alternate connection file DFHMAC1F. This unique tag comprises the CICS applid and the EIBTASKN of DFHMADPL. The processing is as follows:

1. In the first FEPI server adapter that is invoked in the service flow, the logical unit owner is updated in the alternate connection file using the unique tag and the shared user ID and pool name.
2. Any subsequent FEPI server adapter that runs as part of the same BTS process uses the same unique tag identifier to retrieve the correct assigned and in-session logical unit.
3. The assignment remains in effect for the duration of the service flow, which might include the invocation of multiple FEPI server adapters
4. The unique tag allows for running concurrent processes with the same shared user ID and does not change when a single process is running.

If the service flow ends abnormally, the service requester handles the condition by one of the following methods:

- Initiating a service flow to compensate for the failed process
- Issuing a cancel request to release the BTS resources

In addition, the CICS SFR interface program cleans up the assigned logical units if the service flow ends abnormally.

Error handling in synchronous mode

If a service flow ends abnormally, the logical units can remain assigned to the shared user ID. When the service flow completes, no BTS data-container information is available for subsequent users to locate, use, and log off assigned logical units. For this reason, the CICS SFR interface program closes any FEPI conversations and deletes the assignment of logical units, before returning to the service requester, even if the service flow did not complete successfully.

DFHMADPL uses the unique tag that is stored in the DFHMAC1F file to:

- Locate all logical units that are used in the BTS process
- End all the FEPI conversations
- Delete the assigned logical unit from the connection file record
- Initialize the target interaction file record

Error handling in asynchronous mode

If a service flow ends abnormally, the BTS process does not complete and the data-containers are available for subsequent use. To complete the failed BTS process and release the BTS resources, the

service requester must either invoke a compensating flow in the same mode and use shared user IDs, or issue a cancel request to release the BTS resources and complete the process.

If the service requester invokes a compensating flow, the CICS SFR interface program:

- Acquires the failed process using the process name and process type.
- Initiates a new BTS process under which the compensating flow can run.
- Retrieves the information in the data-containers of the failed process.
- Cancels the acquired process, completing the failed BTS process and releasing the resources.
- Put the information into the data-containers associated with the new BTS process.

The unique tag from DFHMAC1F is available to the next service flow and is used to locate all the logical units that were in the failed process. The logical units must be released as part of the following service flow.

If the service requester cancels the failed process, the CICS SFR interface program does the following:

- Acquires the failed BTS process, gaining access to the data-containers that include the unique tag for the shared user ID.
- Uses the unique tag and the user ID to locate all of the logical units that were used in the failed process.
- Ends the FEPI conversation for each logical unit.
- Deletes the record in the connection file for the logical units.
- Initializes the target interaction file. This file contains the last screen buffer that was sent or received for the assigned logical unit.
- Cancels the acquired process, completing the BTS process and releasing BTS resources.

Service flows with different user IDs

When you use different user IDs for service flows, the FEPI logical units are assigned and released as modeled in the service flow project tools.

Leaving a user ID assigned for subsequent processing might improve the performance because the signon processing to access target applications is not required for each service flow request.

If a logical unit is left assigned to a user ID in synchronous or asynchronous mode for use in subsequent process execution, the service requester is responsible for logging off the user ID when it is no longer required. For example, as a normal end of day processing strategy, the service requester could invoke a service flow to locate, log off, and release any assigned logical units. Failure to incorporate a cleanup strategy for a specific user might not leave logical units available over time to other users, unless your configuration is such that a significant number of logical units are available.

The CICS Service Flow Runtime does not locate and clean up logical units that are left assigned upon successful or unsuccessful execution of service flows.

Link3270 server adapter processing

The Link3270 server adapter enables a service requester to conduct an interactive request and reply dialog with 3270 application programs that are running in CICS by using the CICS Link3270 bridge. This interactive dialog can use BMS maps, screen buffers, or a combination of the two.

To perform Link3270 server adapter processing, you must ensure that the CICS region in which CICS Service Flow Runtime is installed is correctly configured. If the target application uses BMS maps, you must load the map sets in a load library in the DFHRPL DD statement or concatenation of the CICS region. You must also define each map set load module in the CICS system definition (CSD) file.

When the Link3270 server adapter is invoked during request processing, it performs the following actions:

- Allocates a Link3270 bridge facility.

- Initiates the CICS target application transaction, using the data from the service requester as input to a map or screen buffer. The Link3270 bridge satisfies the command issued by the target 3270 application.
- Parses the BMS application data structure or 3270 screen buffer that is sent by the CICS target application.
- Identifies the transaction, screen fields, attributes, and data.
- Constructs and sends an appropriate response using the application data structure or screen buffer, based on the modeled flow and on simple business logic.
- Handles the next screen by parsing, identifying, and constructing another vector or keystroke. The Link3270 bridge passes the symbolic map or screen buffer to the Link3270 server adapter, where it can be used for the next transaction or to format and return a response to the service requester.
- Manages state information for the respective CICS user ID and bridge facility.
- Deletes the Link3270 bridge facility.

The Link3270 server adapter begins saving a copy of the application data structure when a **SEND MAP ERASE** command is issued. Additional business data and field attributes, from subsequent **SEND MAP** commands without the ERASE option, are merged with the saved application data structure. The saved application data structure, with the accumulated data, is used in Link3270 server adapter processing. The same processing applies to screen buffers that use a link3270 data stream.

The Link3270 server adapter can move business data from the application data structure or screen buffer to any output data-container, as modeled in the flow. If additional business data is required, the Link3270 server adapter must submit additional transactions to the Link3270 bridge. You might have to supply input data for certain fields in the current application data structure or screen buffer. The Link3270 server adapter can obtain this data from any of its input data-containers or from data collected from previously run target 3270 applications.

Transaction routing

Link3270 server adapters support both dynamic transaction routing and transaction routing. You can define the target CICS application transactions as remote or they can be dynamically routed to remote regions using the CICS dynamic transaction routing facility.

The Link3270 server adapter uses the Link3270 bridge mechanism to run 3270 transactions by linking to the DFHL3270 program and passing a COMMAREA that identifies the transaction to run and the data used by the target CICS application.

The Link3270 server adapter performs the following steps:

1. Runs terminate processing in the current target CICS region, with the bridge facility allocated.
2. Deallocates the currently allocated bridge facility.
3. Allocates a new bridge facility in the target CICS region
4. Runs request processing where the transaction is either a remote CICS system name with a remote name of CMAI, or the local CICS system name with a program name of DFHMALIN in the target CICS region.
5. Runs the target CICS application transaction routed to the target CICS region.

For further information regarding CICS function request shipping (transaction routing method), see CICS function shipping. For programming information about the dynamic transaction routing program, see Writing a dynamic routing program.

For further information about the Link3270 bridge mechanism and its support for these types of intercommunication methods, see Developing for external interfaces and Administering the Link3270 bridge.

Configuring the runtime environment to use transaction routing

You can select transaction routing or dynamic transaction routing for Link3270 server adapter processing.

About this task

The routing region refers to the region where CICS Service Flow Runtime is installed.

The purpose of the CMAI transaction in a transaction routing environment is to retrieve any target CICS application transaction COMMAREA information and any TCTUA information from the CICS region where the Link3270 bridge facility is currently allocated, and to populate that same information in a second CICS region before running the next target CICS application transaction routed to that second CICS region.

If all target CICS application transactions are run locally in the routing region when processing using Link3270 server adapter programs, the transaction and program definitions described below are not necessary. To configure transaction routing, follow these steps:

Procedure

1. Specify a transaction definition in the routing region that points to a remote definition for the transaction CMAI.

   This transaction must be equal to the CICS system name (CONNECTION name) for the remote CICS region where the target CICS application transactions run. Each remote CICS region accessed requires a definition in the routing region.
2. Define the program DFHMALIN and transaction ID CMAI in each remote CICS region where your target CICS application transactions run.
3. Define program and transaction definitions for the program DFHMALIN in the routing region.

   The transaction ID must be equal to the CICS system name (CONNECTION name) of the CICS region, not CMAI. The CMAI transaction definition in the routing region is reserved and used for dynamic transaction routing.
4. To perform transaction routing:
   a) Define a transaction in the routing region for every remote target CICS application region with the following attribute values:

```
TRANSaction: REMOTESystem (remote CICS system name/CONNECTION name)

 REMOTE ATTRIBUTES
    DYnamic       ==> No
    ROutable      ==> No
    REMOTESystem  ==> (remote CICS system name/CONNECTION name)
    REMOTEName    ==> CMAI (CICS Service Flow Runtime Initiate/Terminate transaction ID)
```

   b) If some target CICS application transactions run locally in the routing region, specify a transaction definition where the transaction ID is equal to the local CICS system name (the routing region CONNECTION name) with the program attribute set to DFHMALIN as follows:

```
TRANSaction            : (local CICS system name/CONNECTION name)
PROGram             ==> DFHMALIN
REMOTE ATTRIBUTES
DYnamic             ==> No
ROutable            ==> No
REMOTESystem        ==>
REMOTEName          ==>
```

   c) Define the following attributes on your target CICS application transaction definitions:

```
DYNAMIC attribute = NO
REMOTESYSTEM      = (CICS system name/CONNECTION name)
```

   d) Define the following transaction and program in all target CICS application regions:

```
CMAI      ==> Initiate/Terminate transaction ID
DFHMALIN  ==> Initiate/Terminate program name
```

5. To perform dynamic transaction routing:
   a) Include a transaction definition for the transaction CMAI in the routing region and code your dynamic transaction routing program accordingly.

      You need this definition in addition to the transaction definitions for each target CICS region. In dynamic transaction routing, this definition is used to perform the initiate processing.

      Define the CMAI transaction with the following REMOTE ATTRIBUTES:

```
DYnamic       ==> Yes
ROutable      ==> No
REMOTESystem  ==>
REMOTEName    ==> CMAI (CICS Service Flow Runtime Initiate / Terminate transaction ID)
```

   b) Define the DYNAMIC attribute on your target CICS application transaction definitions as YES.

Facility state cleanup processing

When CICS deletes a facility, the state information associated with that facility might remain.

CICS Service Flow Runtime includes processing that can clean up facility state information for Link3270 server adapters.

The facility and state information referred to in the following processing descriptions refers to business state data saved for processing service flows generated and deployed using the service flow project tools. The data referred to in the following sections is the facility state data that has been collected and saved by the CICS Service Flow Runtime processing, not the data collected using the 3270 LinkBridge.

The CICS Service Flow Runtime supports two types of facility state cleanup processing:

- Facility state cleanup processing for temporary storage queues (TSQs)
- Facility state cleanup processing for VSAM

Facility state cleanup processing — TSQ

If you are running simple, nonpersistent Link3270 service flows, facility state cleanup processing runs against temporary storage queues (TSQ) and is handled by the CICS Service Flow Runtime Facility State Cleanup (TSQ) program (DFHMALSC).

DFHMALSC browses Link3270 facility state temporary storage (TS) queues and initiates the processing to delete the expired CICS Service Flow Runtime Link3270 facility session state data. It invokes the processing to deallocate the associated Link3270 bridge facilities that CICS has not automatically deleted because the facility was inactive for the keeptime interval. See Developing for external interfaces for more information on the keeptime interval processing by CICS.

Processing is as follows:

1. DFHMALSC browses the Link3270 facility state temporary storage (TS) queue.

   The TS queue names are 16 bytes long and of the following format:

```
TSQ name = "DFHMA"  +  facility token (8 byte hex value) +  x'FFFFFF' (3 byte hex value = HIGH-VALUES).
```

2. If the following conditions are true, DFHMALSC calls the Link3270 Facility Deallocate Cleanup program, DFHMALFD:
   - A matching TS queue is found and not in use.

     The CICS Service Flow Runtime facility session state expiration time is checked to see if it has been exceeded.
   - The expiration time of the facility session state has been exceeded.

     The existence of the Link3270 bridge facility is checked and is not in a RELEASED state.
3. If the Link3270 bridge facility has been deleted, the TS queue containing CICS Service Flow Runtime facility session state data is deleted and the browse of the TS queues is resumed. When the 'END' condition is encountered on the browse, this cleanup task is scheduled to be started at the requested

SI interval. See “Configuring the autostart procedure for the Link3270 facility state cleanup programs” on page 29 for information on setting the SI interval.

4. The Link3270 Facility Deallocate Cleanup program deallocates existing bridge facilities and deletes the associated facility session state data whether that data is stored in a TS queue or a VSAM data set.

Facility state cleanup processing — VSAM

The Facility Sate Cleanup (VSAM) program (DFHMALFC) manages facility state cleanup processing when you are running simple, persistent and complex Link3270 service flows.

DFHMALFC browses the Link3270 state file and invokes the processing to delete the expired Link3270 facility session state data. It invokes the processing to deallocate the associated Link3270 bridge facilities that CICS has not automatically deleted because the facility was inactive for the keeptime interval. See Developing for external interfaces for more information on the keeptime interval processing by CICS.

The Link3270 state file used by DFHMALFC is called DFHMAL2F.

Processing is as follows:

1. DFHMALFC browses the Link3270 facility state file.
2. If the following conditions are true, DFHMALFC calls the cleanup program, DFHMALFD:
   - A matching record is found and not in use.

     The facility session state expiration time is checked to see if it has been exceeded.
   - The facility session state expiration time has been exceeded

     The existence of the Link3270 bridge facility is checked and is not in a RELEASED state.
3. If the Link3270 bridge facility has been deleted, the record containing facility session state data is deleted and the browse of the Link3270 State file is resumed. When the 'END' condition is encountered on the browse, this cleanup task is scheduled to be started at the requested SI interval. See “Configuring the autostart procedure for the Link3270 facility state cleanup programs” on page 29 for information on setting the SI interval.
4. The cleanup program deallocates existing bridge facilities and deletes the associated facility session state data whether that data is stored in temporary storage or a VSAM data set.

Managing shared temporary storage queues in a multiregion environment

In a multiregion environment, where an application is using shared temporary storage queues, you have extra considerations when using the Link3270 bridge facility because every CICS region has a unique Link3270 bridge facility.

About this task

If you form all or part of the name of shared temporary storage queues from the terminal ID, you must use the same terminal ID for each Link3270 bridge facility in a pseudoconversation across the CICS regions. You can use a bridge facility autoinstall program exit for this purpose. The CICS system initialization parameter **AIBRIDGE** controls the calling of a bridge facility autoinstall user replaceable module.

Procedure

1. Change the AIBRIDGE parameter to AIBRIDGE=YES.
2. Modify the CICS-supplied sample autoinstall user replaceable module, DFHZATDX, to change the terminal ID that is supplied by the Link3270 bridge.

   The sample code below changes the last character of the terminal ID from a } to a #.

```
INSTALL_BRIDGE_FACILITY DS 0H
         USING INSTALL_BRFAC_COMMAREA,R2 Address commarea
* ==> PUT INSTALL CODE HERE
USESEL   DS 0H
```

```
*
*     This sample accepts the selected termid/netname.
*     Special consideration MUST be given to how this termid
*     will be used.
*     In particular it must not conflict with the namespace of
*     real terminals.
*
         L     R5,INSTALL_BRFAC_SELECTED_PTR
         USING INSTALL_BRFAC_SELECTED_PARMS,R5
         L     R8,INSTALL_BRFAC_TERMID_PTR
         MVC   SELECTED_BRFAC_TERMID,0(R8)
         L     R8,INSTALL_BRFAC_NETNAME_PTR
         MVC   SELECTED_BRFAC_NETNAME,0(R8)
*     following 5 lines inserted for application shared TSQ's
         CLI   SELECTED_BRFAC_TERMID+3,X'D0' is the last char a }?
         BNE   RETURN   If not then already altered, so accept it
*     otherwise change the last char of the termid and netname
         MVI   SELECTED_BRFAC_TERMID+3,X'7B'     change } to  #
         MVI   SELECTED_BRFAC_NETNAME+3,X'7B'    change } to  #
         MVI   SELECTED_BRFAC_RETURN_CODE,RETURN_OK Say all OK
*
         B     RETURN              EXIT PROGRAM
```

3. If you want to write your own autoinstall user replaceable module, edit the system initialization parameter **AIEXIT** to specify the name of the module.

What to do next

For more information about DFHZATDX, see Developing for external interfaces and Customizing with user-replaceable programs in Developing system programs.

Managing state cleanup for Link3270 server adapters

The Link3270 bridge facility that is allocated in the initial region at the start of the pseudoconversation is referred to as the primary Link3270 bridge facility. Do not delete the application temporary storage queues until the completion of the pseudoconversation, at which time the primary Link3270 bridge facility is deleted.

About this task

The XFAINTU global user exit runs when the Link3270 bridge facility expires, so you can add your own processing to this user exit to ensure that the temporary storage queues are not deleted until the pseudoconversation is complete.

When the primary Link3270 bridge facility is allocated, CICS SFR writes a temporary storage queue record containing the primary bridge facility token, using the terminal ID as a portion of the queue name. Use the provided sample to check if the expired bridge facility is the primary facility. You must write a program that deletes the temporary storage queues, which is invoked by XFAINTU if it matches the primary facility token with the expired Link3270 bridge facility.

Procedure

1. Write a program to delete the application temporary storage queues.
2. Use the provided sample global user exit, editing the following command:

   ```
   EXEC CICS LINK PROGRAM(deletets)
   ```

   The value of *deletets* must be the name of the program that deletes the queues associated with the bridge facility terminal ID.
3. Enable the XFAINTU global user exit in CICS.

Results

When the bridge facility expires in the CICS region, the XFAINTU global user exit runs. It reads the temporary storage queue record and compares the primary facility token to the bridge facility token

passed to the XFAINTU global user exit when it was first allocated. If they match, the temporary storage queue is deleted by the XFAINTU global user exit using the specified program.

Example

Here is an example of the XFAINTU global user exit:

```
DFHUEXIT TYPE=EP,ID=XFAINTU Standard UE parameters for XFAINTU 45300000

*********************************************************************
*  REGISTER USAGE :                                                 *
*  R0 -                                                             *
*  R1 - address of DFHUEPAR on input, and used by XPI calls         *
*  R2 - address of standard user exit parameter list, DFHUEPAR      *
*  R3 - BASE address                                                *
*  R4 - address of XFAINTU request byte                             *
*  R5 - address of XFAINTU TIDY-UP TYPE                             *
*  R6 - work register                                               *
*  R7 - ADDRESS OF TS QUEUE SUFFIX TABLE                            *
*  R8 - ADDRESS OF TCTUA                                            *
*  R9 - ADDRESS OF BRIDGE FACILITY NAME                             *
*  R10-                                                             *
*  R11- ADDRESS OF EIB                                              *
*  R12-                                                             *
*  R13- address of kernel stack prior to XPI CALLS                  *
*  R14- used by XPI calls                                           *
*  R15- return code, and used by XPI calls                          *
*  (The register equates are declared by the DFHUEXIT call above)   *
*********************************************************************
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 2
XFAINTU  DFHEIENT DATAREG=13,CODEREG=3,EIBREG=11
XFAINTU  AMODE 31
XFAINTU  RMODE ANY
         LR    R2,R1              Address standard parameters
         USING DFHUEPAR,R2
         L     R9,UEPFANAM  ADDRESS OF BRIDGE FACILITY NAME
         MVC   BFTRMID(4),0(R9)    GET TERMID
         L     R4,UEPFAREQ  Address of why exit called: Init or Tidy-up
         L     R5,UEPFATUT  Address of XFAINTU Tidy-up type
         L     R8,UEPFAUAA         TCTUA address
         L     R6,UEPFATK          LOAD Facility Token address
         MVC   BFAC(8),0(R6)       get bridge facility token
*********************************************************************
* WHY WAS EXIT CALLED, INITIALIZATION OR TIDY-UP?                   *
*********************************************************************
         CLI   0(R4),UEPFATU
         BE    TIDYUP
*********************************************************************
* INSERT INITIALIZATION for TCTUA HERE                              *
*********************************************************************
         B     END

TIDYUP   DS    0H
*********************************************************************
* CHECK IF PRIMARY FACILITY                                         *
*********************************************************************
         EXEC  CICS READQ TS QNAME(QNAME),                          +
               INTO(PBFAC),                                         +
               ITEM(1),                                             +
               LENGTH(PBFACLEN),                                    +
               RESP(RESPCD).

         CLC   RESPCD,DFHRESP(NORMAL)
```

```
         BNE   END
         CLC   BFAC(8),PBFAC
         BNE   END

*********************************************************************
* Primary facility is being deleted, so delete TS queues           *
*********************************************************************
*  DELETE APPLICATION TS QUEUES WITH APPLICATION PROGRAM
         EXEC  CICS LINK PROGRAM(deletets),
                         COMMAREA(BFTRMID),
                         LENGTH(4),
                         RESP(RESPCD).

END      DS    0H
GLUEND   DS    0H                 Standard GLUE ending code
         LA    R15,UERCNORM       CONTINUE PROCESSING
         DFHEIRET RCREG=15        Return to CICS

*********************************************************************
*        Constants                                                  *
*********************************************************************
QNAME    DS    0H
QNAMEPRE DC    CL5'DFHMA'
BFTRMID  DS    CL4
SUFFIX   DS    XL7'FFFFFFFFFFFFFF'
PBFACLEN DC    H'8'
PBFAC    DS    CL8
BFAC     DS    CL8
RESPCD   DS    F
         DFHEISTG
         END XFAINTU
```

Processing of shared and unique user IDs

A shared user ID is not associated with a single user, but instead more than one user at a time can use it. You are recommended to use unique user IDs so that authentication can be based on user identity and to help with problem diagnosis.

If you use unique user IDs, Link3270 bridge facilities and the associated session state data can be used in one request, left assigned to the unique user ID, and reused in subsequent request processing. The application can be left at a particular screen as modeled.

If you use shared user IDs in a Link3270 service flow, there is no impact on the management of the business state file, DFHMAL2F.

Authentication using unique user IDs

If you require authentication, you can configure the WebSphere MQ-CICS bridge to use a unique user ID and optionally provide a password for authentication by a supported external security manager such as RACF.

You can pass user IDs to the CICS Service Flow Runtime as part of a request message in the WebSphere MQ MQMD structure, field name MQMD-USERIDENTIFIER. See Developing applications in IBM MQ product documentation for specific information on MQMD-USERIDENTIFIER.

This user ID is used in security authentication in the WebSphere MQ-CICS bridge if configured appropriately. You can implement different levels of authentication using a user ID and optionally provide a password. See “Considering security” on page 19 for more information.

Link3270 bridge restrictions

The Link3270 bridge is supported on every release of CICS Transaction Server. However, support for the Link3270 bridge CICS API varies, depending upon the version and release of CICS Transaction Server.

Although CICS Service Flow Runtime has a prerequisite to run in CICS Transaction Server Version 3.2, no such prerequisite applies for the target CICS application regions where your user transactions and target applications are running. If CICS Service Flow Runtime is running user transactions using the Link3270 bridge and if those user transactions are running in CICS Transaction Server for z/OS 2.2, CICS Service Flow Runtime and those user transactions are restricted to the CICS Transaction Server for z/OS 2.2 Link3270 bridge supported CICS API.

The restrictions table, DFHEIDBR, identifies commands that might not be supported when the generated Link3270 server adapter uses the Link3270 bridge. The table is in the CICS sample library *hlq*.SDFHSAMP. Use this table with an IBM-supplied load module scanner utility, DFHEISUP, to analyze your user transactions and target applications.

The following list provides further restrictions when using the Link3270 bridge with CICS Service Flow Runtime.

- User transactions cannot issue an **EXEC CICS START** command to start other local or remote user transactions. The target applications must use **EXEC CICS RETURN TRANSID() IMMEDIATE** instead.
- User transactions resulting in outbound Link3270 bridge vectors that exceed 32 000 bytes are not supported.

  Link3270 server adapters currently support only one outbound vector with a maximum length of 32 000 bytes. The COMMAREA length used to LINK to the Link3270 bridge router program DFHL3270 is 32 000 bytes.
- Link3270 server adapters do not support BRIHT-GET-MORE-MESSAGE and BRIHT-RESEND-MESSAGE features of the Link3270 bridge.
- Link3270 server adapters do not support the PAGING option on the **EXEC CICS SEND** command.
- All interactions with target applications are performed using the Link3270 bridge session mode.
- You must use an ADS descriptor in mapset load modules. If you do not, CICS SFR issues the DFHMA07018E message.

Link3270 server adapter data-containers

Link3270 data-containers are used to store state and status data and inputs and outputs for the Link3270 server adapter program.

The following tables list the Link3270 data-containers that are used during request processing of simple and complex service flows.

Table 6. Link3270 server adapter data-containers for simple service flows

Name	Size (bytes)	Contents and usage
ADAPTER.OUTPUT	Variable up to a maximum of 32 760	Output from the Link3270 server adapter in a simple service flow.
ADAPTER.STATUS	512	Status information for the Link3270 server adapter in a simple flow, including: • Interaction status • Error information
LINK3270.INPUT	Variable up to a maximum of 32 760	Mapped input data.
LINK3270.STATE	Variable up to a maximum of 32 760	State information for the Link3270 server adapter.

Table 7. Link3270 server adapter data-containers for complex service flows

Name	Size (bytes)	Contents and usage
ADAPTER.SHARED.C	Compiler limit for IBM COBOL	The application context that is shared between Link3270 server adapters. This data-container ensures that data is kept consistent between multiple Link3270 server adapters in a complex service flow.
LINK3270.INPUT	Variable up to a maximum of 32 760	Mapped input data.

Table 7. Link3270 server adapter data-containers for complex service flows (continued)		
Name	**Size (bytes)**	**Contents and usage**
LINK3270.OUTPUT	Variable up to a maximum of 32 760	Output from the Link3270 server adapter.
NAVIGATOR.STATUS	512	Status information for the Link3270 server adapter, including: • Interaction status • Error information
LINK3270.STATE	Variable up to a maximum of 32 760	State information for the Link3270 server adapter.

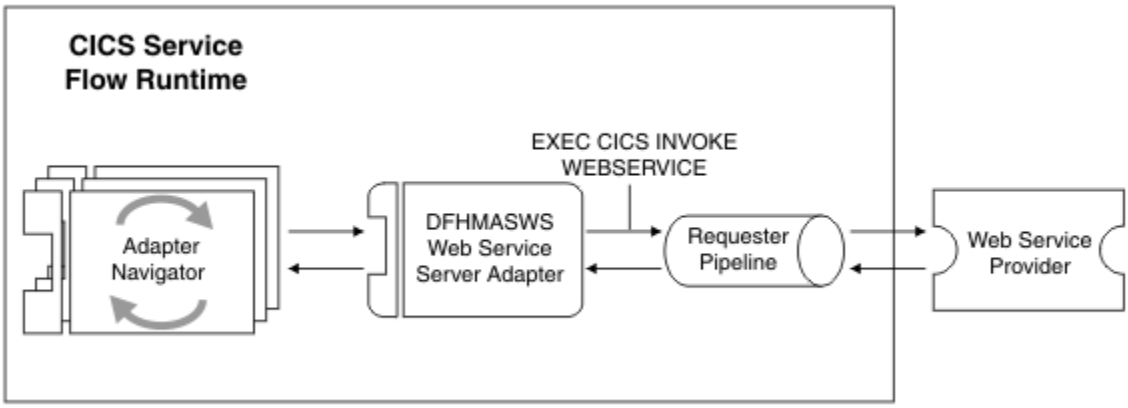
Web services server adapter processing

The Web service server adapter is called DFHMASWS and runs by default under the CMAO transaction. You must define a suitable PIPELINE and WEBSERVICE resource in the CICS region, so that DFHMASWS can send outbound web service requests.

When this server adapter is invoked during request processing, the details of the Web service request and data are passed in BTS data-containers from the generated Adapter Navigator to DFHMASWS. DFHMASWS uses this data to send the Web service request by issuing the **EXEC CICS INVOKE WEBSERVICE** command. This request is processed in a requester mode pipeline and sends a SOAP message to the designated Web service provider. The Web service provider might be on another CICS system or an external provider.

When a response message is received, the pipeline processes it and returns it to the Web service adapter in one or more containers. The Web service server adapter then passes the response message data back to the Adapter Navigator using a data-container.

This process is shown in the figure below.



The data-containers that are passed to DFHMASWS store the data that is specified by service flow project tools when the web service request is modeled. The COMMAND.INPUT data-container contains the data to be sent to the Web service provider. The WEBSERVICE.DATA data-container stores the information regarding the target Web Service.

When a response is received from the Web service provider, DFHMASWS updates the WEBSERVICE.DATA data-container to specify whether the response was successful or a SOAP fault was received.

- If a successful response is received, DFHMASWS returns it in the COMMAND.OUTPUT data-container.
- If a SOAP fault is received, the details are placed in the WEBSERVICE.DATA data-container and the DFHMASWS server adapter takes the action that was modeled in the service flow.

Queue server adapter processing

The Queue server adapter is called DFHMASCQ and runs under the CMAU transaction. It is invoked only in complex flows and is therefore always invoked by a flow navigator. This server adapter handles the requests and responses for WebSphere MQ-enabled target applications.

To use this server adapter, you must correctly configure the CICS region to use the CICS-WebSphere MQ adapter. The CICS-WebSphere MQ adapter connects a CICS region to a WebSphere MQ queue manager, enabling CICS application programs such as DFHMASCQ to use the WebSphere MQ Interface (MQI).

The flow navigator passes COMMAND.INPUT and QUEUE.DATA BTS data-containers to the Queue server adapter. The COMMAND.INPUT data-container contains the application data that must be sent to the target application. The QUEUE.DATA data-container contains the details of the type of WebSphere MQ request that is required. The request is either an **MQGET** call or an **MQPUT** call.

DFHMASCQ retrieves the details of the local queue that is specified in the service flow from the service flow BTS data-containers to send a WebSphere MQ request to that local queue. The service flow project tools allows you to make three types of requests to WebSphere MQ:

- DATAGRAM
- REQUEST

If the service flow uses DATAGRAM, the Queue server adapter performs an **MQPUT** command and does not wait for a defined reply queue or a response to the **MQPUT** command. The WebSphere MQ-enabled application that is the target of the WebSphere MQ call might be the service requester that issued the original request or an application that performs some work as part of the server adapter processing. Whatever role the WebSphere MQ-enabled application plays, it does not have to issue any response to the server adapter. If the target application is the service requester, the flow navigator processes the **MQPUT** command as an early reply to the request.

If the service flow uses REQUEST, the Queue server adapter performs an **MQPUT** command and returns to the flow navigator. The flow navigator invokes the Queue server adapter again to perform an **MQGET** command to retrieve the response from the specified reply queue.

Managing state information

The CICS Service Flow Runtime manages the state information to support the work that satisfies one invocation. *State information* refers to the business request state data and information.

About this task

After the invocation is satisfied, state information is not retained or maintained across multiple invocations of CICS Service Flow Runtime. In the case of a failure, CICS Service Flow Runtime retains the state information and application data for a subsequent service flow.

The way in which CICS Service Flow Runtime manages state information for Link3270 bridge server adapters varies according to whether the service flow is complex or simple and if the service flow is persistent.

Business state data management in persistent service flows

For both persistent and nonpersistent complex service flows and for simple persistent service flows, CICS SFR stores, retrieves, and deletes Link3270 server adapter business state data in a CICS VSAM file, DFHMAL2F.

The VSAM file is used to manage the business state data of facilities and ownership of facilities left allocated by service flow processing. Normally, this file stores the last bridge vector received (maximum 32 000 bytes ADS, text, or 3270 datastream data) from the defined target CICS application transaction as modeled in the service flow. The vector data is used in subsequent Link3270 server adapter processing to determine the allocated facility business state; for example, the last CICS transaction run, the last **BMS SEND MAP** application data structure (ADS), the last BMS mapset and map names.

The are two VSAM records for each allocated Link3270 bridge facility are as follows:

- The first VSAM record contains the Link3270 facility business state information used in Link3270 server adapter processing.
- A second VSAM record contains any Link3270 facility business state text information used in Link3270 server adapter processing as the result of a **BMS SEND TEXT** command or **SEND** command containing textual information.

  In this record, the key field LS-KEY-FILLER contains a value of 'TEXT".

Based on the setting of the deallocate facility indicator in the Link3270 server adapter and the completion status of that adapter, the facility might remain allocated and its associated business state retained when the adapter server completes its processing. If so, the Link3270 server adapter business state file record is written containing the following:

- Allocated facility business state information for that facility owner
- Allocated Link3270 bridge facility token
- The service name that is defined in the service flow properties file.

The facility and its associated business state is available for subsequent use by the owner of that facility in another task. The owner is the signed-on user ID to the local CICS region, as determined by an **EXEC CICS ASSIGN** command.

When Link3270 server adapter processing is complete, if the Link3270 bridge facility is left allocated with its associated facility business state data stored in the facility business state file (DFHMAL2F), the Link3270 facility state token is returned to the service requester in the reply message for use in subsequent request processing if required. See “DFHMAH header structure” on page 47 for further information on the meaning and use of DFHMAH header structure field, DFHMAH-STATETOKEN.

If a Link3270 facility state token is left blank in the request message and a Link3270 server adapter is invoked, a new Link3270 facility is allocated and used in Link3270 server adapter processing.

In addition, Link3270 facilities and their associated business state data might be deallocated and deleted, respectively, by the system cleanup tasks. See “Facility state cleanup processing” on page 71 for a description of how CICS SFR cleans up the facility business state VSAM file.

Also, as a normal end-of-day processing strategy, the service requester might invoke a modeled flow to locate any allocated facility, deallocate the facility, and delete its associated facility business state data. This flow must run for each allocated Link3270 facility state token by each service requester invoking service flows.

Business state data management in nonpersistent service flows

In a simple, nonpersistent service flow, the Link3270 server adapter business state data is stored, retrieved, and deleted in a multiple item CICS temporary storage queue (TSQ).

The TSQ has up to two items for each allocated Link3270 bridge facility:

- The first item contains the Link3270 facility business state information used in Link3270 server adapter processing.
- The second item contains any Link3270 facility business state text information used in Link3270 server adapter processing as the result of a **BMS SEND TEXT** command or **SEND** command containing textual information.

The TSQ is used to manage the business state data of facilities and ownership of facilities left allocated. Normally, the TSQ stores the last bridge vector received (maximum 32 000 bytes ADS , text, or 3270 datastream data) from the defined target CICS application transaction as modeled in the service flow. The vector data is used in subsequent Link3270 server adapter processing to determine the allocated facility business state, for example, the last CICS transaction run, the last **BMS SEND MAP** application data structure (ADS), the last BMS mapset and map names.

Based on the setting of the deallocate facility indicator in the Link3270 server adapter and the completion status of that adapter, the facility might remain allocated and its associated business state retained when

the adapter server completes its processing. If so, the Link3270 server adapter business state TSQ is written containing the allocated facility business state information for that facility owner. The facility and its associated business state is available for the owner of that facility in subsequent request processing. The owner is the signed-on user ID to the local CICS region, as determined by an **EXEC CICS ASSIGN** command.

When Link3270 server adapter processing is complete, if the Link3270 bridge facility is left allocated with its associated facility business state data stored in the facility business state TSQ, the Link3270 facility state token is returned to the service requester in the reply message in the DFHMAH header structure field, DFHMAH-STATETOKEN, for use in subsequent request processing. See "DFHMAH header structure" on page 47 for further information on the meaning and use of DFHMAH header structure field, DFHMAH-STATETOKEN.

If a Link3270 facility state token is left blank in the request message and a Link3270 server adapter is invoked, a new Link3270 facility is allocated and used in Link3270 server adapter processing.

An allocated Link3270 facility might be deallocated and its associated facility business state data deleted when Link3270 server adapter processing is complete, based on the setting of the deallocate facility indicator, MP-BR-DEALLOCATE-IND, for that Link3270 server adapter and the completion status of that Link3270 server adapter.

In addition, Link3270 facilities and their associated business state data might be deallocated and deleted, respectively, by the system cleanup tasks. See "Facility state cleanup processing" on page 71 for a description of how this cleanup is performed.

Also, as a normal end-of-day processing strategy, the service requester might invoke a service flow to locate any allocated facility, deallocate the facility, and delete its associated facility business state data. This flow must run for each allocated Link3270 facility state token by each service requester invoking service flows.

XML request and response processing

You are recommended to use IBM Developer for Z or the CICS Web services assistant as the strategic method for parsing modeled requests. This approach provides XML parsing and generation for both the request header and body.

The CICS Web services assistant is described in detail in The CICS web services assistant. For information on using the tooling to perform XML processing, see the IBM Developer for Z help that is provided with the tool.

For backwards compatibility, CICS Service Flow Runtime provides an internal XML parsing function, which is nonstrategic.

XML request and response support consists of two main functions:

1. The XML parsing function parses an inbound XML request message and maps XML elements to a fixed format COMMAREA. See "XML message formats" on page 159 for a sample of a request message in XML format.
2. The XML generation function generates an XML response message from a fixed format COMMAREA.

In the runtime environment, the processing associated with XML request and response messages varies according to the way the service requester incorporates the XML into the request message. The request message can be divided into two main parts:

- The message header DFHMAH
- The application data

For more information, see "Sending the request message in a COMMAREA" on page 42.

The following table indicates the message composition formats for XML requests and whether or not the runtime environment supports the message composition format:

Table 8. Supported XML message composition	
XML location	**Supported**
In the message header and in the application data	Yes
In the application data only	Yes
In the message header only	No

See "XML message formats" on page 159 for samples of the XML that can be used in request and reply messages.

XML request processing

At run time, the CICS SFR interface program DFHMADPL initiates XML request processing.

When DFHMADPL determines that the request message header is in XML, the following processing occurs:

1. DFHMADPL issues a call to the XML to COBOL Converter program DFHMAXMI.
2. The program DFHMAXMI converts the header data from XML into a COBOL header format.

   DFHMAXMI saves the XML declaration portion of the XML and passes it back to DFHMADPL. This information is used for XML response processing.

   DFHMAXMI reports any errors it encounters to the CICS system log using the message identifier of IGZ0280S. The CICS error information is returned to the service requester.
3. If the application data portion of the request message is also XML, DFHMADPL reads the service flow properties file for the name of a user-defined application data XML converter program that converts XML data into the COBOL data structure. This converter program is defined in the model of the service flow and is saved in the service flow properties file.

If the properties of the service flow do not include a value for an XML converter program, the application data in the request message is not converted.

If the program call issued by DFHMADPL to the user-defined converter program fails, CICS Service Flow Runtime returns an error message to the service requester.

XML response processing

At run time, XML response processing is initiated when the server adapter passes the reply data from the CICS target application to the CICS SFR interface program DFHMADPL.

DFHMADPL constructs a reply consistent with the format of the request message. When the request message is an XML document that is, the header is in XML format), the reply is a complete XML document. When only the application data is XML, the reply is constructed with a standard format header and the application data as an XML document.

DFHMADPL constructs a response by performing the following steps:

1. Invokes the COBOL to XML converter program to convert the application data. The purpose of this program is to place the proper XML tags around the application data that resides in the response message. For example:

   ```
   <dfhmaad>
   Application data from response message
   </dfhmaad>
   ```

   The COBOL to XML converter program is a user-defined program that you define in the model of the service flow. If you do not provide a converter program for the reply message, the application data is returned in COBOL. If DFHMADPL fails to invoke the converter program, an error is returned to the service requester.
2. Calls the COBOL to XML converter program DFHMAXMO. DFHMAXMO performs the following steps:

a. Converts the structure of the header in the response message from COBOL to an XML format.

b. Places the XML declaration at the top of the XML message. The XML declaration was saved by the converter program DFHMAXMI on the inbound request.

c. Takes the reply data formatted by the user-defined COBOL to XML converter program and converts it to XML.

d. Reports any errors it encounters to the CICS system log, under the CEEMSG section. An error is returned to the service requester.

Interface to the XML header converter program DFHMAXMI

The CICS SFR interface program invokes a user-defined XML data to COBOL conversion program through a standard interface that is based on the *XML Enablement of the Enterprise* feature in the IBM Developer for Z tool.

```
WORKING-STORAGE SECTION.
 .
 ,
 01 APPLICATION-DATA               PIC X(32768) VALUE SPACES.
 01 X-XML-INT-LEN                PIC 9(9) BINARY.
 01 X-XML-INT-TXT                PIC X(32768).
 01 X-CONVERTER-RETURN-CODE       PIC S9(9) BINARY.
 01 USER-PARSER-PROGRAM           PIC X(08) VALUE SPACES.
 .
 .
 PROCEDURE DIVISION.
 .
 .
 MOVE user-program TO USER-PARSER-PROGRAM.
CALL USER-PARSER-PROGRAM  USING APPLICATION-DATA
                                X-XML-INT-LEN
                                X-XML-INT-TXT
                                OMITTED
                    RETURNING
                                X-CONVERTER-RETURN-CODE
```

Figure 6. Call to user-defined converter program

Error processing

CICS Service Flow Runtime writes all errors that occur when processing service flows to a transient data queue (TDQ) called CMAC. This queue is defined as an alias of the CSMT TDQ during the setup of CICS SFR. All CICS SFR messages have a prefix of DFHMA and appear next to CICS messages.

If an error occurs during request processing, CICS SFR writes a message describing the error to the TDQ. Each error message contains a standard set of information, including the user ID, transaction, request name, and program. Depending on the type of message, additional information might be present in the error to assist you in diagnosing the problem.

In addition, the error processing writes out trace point AP00067 when an error occurs, to provide you with additional diagnostic information. Each message number has a resource ID. The prefix of the resource ID is SFR, followed by the message number. For example, if message DFHMA01001E is written to the TDQ during error processing, a trace point with resource ID SFR01001E is also issued.

Errors that occur outside the control of the service flow are not written to the CMAC TDQ. For example, the following types of errors are not written to the TDQ:

- Errors in the WebSphere MQ or CICS environments that occur outside of the CICS SFR environment. These errors are logged in the WebSphere MQ or CICS environments.
- Errors in CICS applications.
- Errors in WebSphere MQ-enabled applications that are invoked by the Queue server adapter.
- Errors in target application programs that are invoked by the Program link server adapter.

If CICS SFR cannot write a message to the TDQ, it writes a message to the console to inform you that it was unable to access the CMAC TDQ.

BTS data-containers

During request processing, BTS maintains the data-containers that are used to pass data between activities or between different invocations of the same activity.

Each data-container is associated with an activity or process; it is identified by its name and by the activity for which it is a container. Data-containers are recoverable resources. They are written to disk and are restored at system restart.

Process data-containers

A process data-container is associated with the root activity of the BTS process. The CICS SFR interface program DFHMADPL creates the BTS process to invoke a deployed service flow. All the activities that are part of the BTS process can read process data-containers, but only DFHMADPL can write to and update them.

The following table lists the process data-containers.

Table 9. Process data-containers

Name	Size (bytes)	Contents
ADAPTER.PROCESS	1 332	This data-container contains: • The message header structures from the request message • The status of the Navigation Manager and service flow
ADAPTER.INPUT	Variable up to a maximum of 32 376	This data-container contains the application data from the request message.

Service flow program data-containers

A service flow program data-container is associated with the service flow processing. It can be read and updated by the owning BTS activity, the activity's parents, or by a program that has acquired the activity. The Navigation Manager and flow navigators use these data-containers to store state, status, and other information required by the invoked service flow.

The following table lists the data-containers that are associated with an invoked service flow program.

Table 10. Service flow program data-containers

Name	Size (bytes)	Contents and usage
ADAPTER.OUTPUT	Variable up to a maximum of 32 760	Application data that is sent in the adapter reply message. In synchronous mode, the maximum reply length is 32 376 bytes. The Navigation Manager writes a copy of this data-container to return the reply message
ADAPTER.STATUS	512	Status information: • Detailed service flow processing status • Error information
ADAPTER.LOCAL.C	Compiler limit for IBM COBOL	Application context that is private to the service flow BTS activity. This data-container is used during asynchronous mode only.
ADAPTER.ITERATE	Compiler limit for IBM COBOL	Application work area required for iterative processing. This data-container is used during asynchronous mode only.

Program link server adapter data-containers

The program link server adapter, DFHMASDP, stores its state, processing status, inputs, and outputs in BTS data-containers.

The program link server adapter uses the **EXEC CICS LINK** command, and passes either a COMMAREA that can be up to 32 767 bytes in length to the target application, or a channel and containers.

The following table lists the standard set of DFHMASDP data-containers. Additional data-containers might be made available to DFHMASDP for its processing, if the program link is passing a channel and containers.

Table 11. DFHMASDP data-containers

Name	Size (bytes)	Contents and usage
DPL.DATA	40 006	Contains the programming link to the target application. If the link is using a channel and containers, this data-container also stores the channel name and the number and name of the containers that must be passed in the channel.
COMMAND.INPUT	Variable with a maximum of 32 760	Input data from the flow navigator.
COMMAND.OUTPUT	Variable with a maximum of 32 760	Output from DFHMASDP.
COMMAND.STATUS	256	Status information for DFHMASDP, including: • Interaction status • Error information

FEPI server adapter data-containers

FEPI server adapters use data-containers to store state and status data and inputs and outputs for the FEPI request processing.

The following tables list the FEPI data-containers that are used during request processing of simple and complex service flows.

Table 12. FEPI server adapter data-containers for simple flows

Name	Size (bytes)	Contents and usage
ADAPTER.OUTPUT	Variable up to a maximum of 32 760	It contains output from the FEPI server adapter.
ADAPTER.STATUS	512	It contains status information for the FEPI server adapter, including: • Interaction status • Error information
FEPI.INPUT	Variable up to a maximum of 32 760	Mapped input data.
FEPI.STATE	6 144	Target and node state information for the FEPI server adapter. This data-container is used only when a FEPI server adapter error occurs.

Table 13. FEPI server adapter data-containers for complex flows		
Name	**Size (bytes)**	**Contents and usage**
ADAPTER.SHARED.C	Compiler limit for IBM COBOL	It contains the application context that is shared between FEPI server adapters. This data-container ensures that data is kept consistent between multiple FEPI server adapters in a complex service flow.
FEPI.INPUT	Variable up to a maximum of 32 760	Mapped input data.
FEPI.OUTPUT	Variable up to a maximum of 32 760	It contains output from the FEPI server adapter.
FEPI.STATE	6 144	Target and node state information for the FEPI server adapter. This data-container is used only when a FEPI server adapter error occurs.
NAVIGATOR.STATUS	512	It contains status information for the FEPI server adapter, including: • Interaction status • Error information

Link3270 server adapter data-containers

Link3270 data-containers are used to store state and status data and inputs and outputs for the Link3270 server adapter program.

The following tables list the Link3270 data-containers that are used during request processing of simple and complex service flows.

Table 14. Link3270 server adapter data-containers for simple service flows		
Name	**Size (bytes)**	**Contents and usage**
ADAPTER.OUTPUT	Variable up to a maximum of 32 760	Output from the Link3270 server adapter in a simple service flow.
ADAPTER.STATUS	512	Status information for the Link3270 server adapter in a simple flow, including: • Interaction status • Error information
LINK3270.INPUT	Variable up to a maximum of 32 760	Mapped input data.
LINK3270.STATE	Variable up to a maximum of 32 760	State information for the Link3270 server adapter.

Table 15. Link3270 server adapter data-containers for complex service flows		
Name	**Size (bytes)**	**Contents and usage**
ADAPTER.SHARED.C	Compiler limit for IBM COBOL	The application context that is shared between Link3270 server adapters. This data-container ensures that data is kept consistent between multiple Link3270 server adapters in a complex service flow.
LINK3270.INPUT	Variable up to a maximum of 32 760	Mapped input data.

Table 15. Link3270 server adapter data-containers for complex service flows (continued)		
Name	**Size (bytes)**	**Contents and usage**
LINK3270.OUTPUT	Variable up to a maximum of 32 760	Output from the Link3270 server adapter.
NAVIGATOR.STATUS	512	Status information for the Link3270 server adapter, including: • Interaction status • Error information
LINK3270.STATE	Variable up to a maximum of 32 760	State information for the Link3270 server adapter.

Queue server adapter data-containers

The queue server adapter, DFHMASCQ, stores its state, processing status, inputs, and outputs in BTS data-containers.

The following table lists the data-containers that are used during request processing by the queue server adapter.

Table 16. Queue server adapter data-containers		
Name	**Size (bytes)**	**Contents and usage**
COMMAND.INPUT	Variable up to a maximum of 32 760	Mapped input data.
COMMAND.OUTPUT	Variable up to a maximum of 32 760	Output from the server adapters.
COMMAND.STATUS	256	Status information from the server adapter; for example: • Interaction status • Error information
QUEUE.DATA	28	The WebSphere MQ request details, including whether the request is a PUT or a GET.

Web service server adapter data-containers

The Web service server adapter, DFHMASWS, stores its state, processing status, inputs, and outputs in BTS data-containers.

The following table lists the data-containers that the Web services server adapter uses during request processing.

Table 17. Web service server adapter data-containers			
Name	**Type**	**Size (bytes)**	**Contents and usage**
WEBSERVICE.DATA	DATA	6847	Web service request that is passed to the requester pipeline.
COMMAND.INPUT	INPUT	Variable up to a maximum of 32 760	Mapped input data.
COMMAND.OUTPUT	OUTPUT	Variable up to a maximum of 32 760	Output from the server adapter.

Table 17. Web service server adapter data-containers (continued)			
Name	**Type**	**Size (bytes)**	**Contents and usage**
COMMAND.STATUS	STATUS	256	Status information from the server adapter; for example: • Interaction status • Error information

Error and journaling data-containers

Error processing and journaling might be necessary at different stages of runtime processing. The error and journaling data-containers might be used in the request processing of all the supported server adapters.

Table 18. Error and journaling data-containers		
Name	**Size (bytes)**	**Contents**
ADAPTER.ERROR	256	Detailed error information that can be used in problem determination.
ADAPTER.JOURNAL	4 MB	Application data modeled at build time. This data-container is used in recovery by the service flow. The Navigation Manager writes a copy of the activity journal container for recovery.

Chapter 9. Troubleshooting and support

This information helps you to diagnose problems with the runtime environment, including help on how to identify the source of errors using available diagnostic facilities, instructions for searching knowledge bases, getting fixes and support, and the process for applying APARs.

Remember that a service flow runs as a CICS BTS application. This means that you can use CICS BTS utilities to produce diagnostic information. Make sure that you have access to the CICS documentation to assist you with problem determination:

- Troubleshooting and support for the following information:
  - Using CICS debugging tools, trace, and dump.
  - CICS BTS messages, trace and dump.
  - CICS Front End Programming Interface (FEPI) messages, trace and dump.
- Developing for external interfaces for detailed information about the Link3270 bridge mechanism.
- CICS messages for information on abend codes and CICS system messages.

Make sure that you use the correct documentation for the level of CICS that your site is running and for other products that you are using with your applications. The wrong level of information might hinder problem determination.

Learning more

The first step in the troubleshooting process is to learn more about the problem symptoms.

The following topics can help you to acquire the background information that you need to effectively troubleshoot problems with CICS Service Flow Runtime.

About troubleshooting

Troubleshooting is a systematic approach to solving a problem.

The first step in the troubleshooting process is to describe the problem completely. Without a problem description, neither you nor IBM can know where to start to find the cause of the problem. This step includes asking yourself basic questions, such as:

- “What are the symptoms of the problem?” on page 89
- “Where does the problem occur?” on page 90
- “When does the problem occur?” on page 90
- “Under which conditions does the problem occur?” on page 90
- “Can the problem be reproduced?” on page 91

The answers to these questions typically lead to a good description of the problem, and that is the best way to begin problem resolution.

What are the symptoms of the problem?

When starting to describe a problem, the most obvious question is "What is the problem?" This question might seem straightforward; however, you can break it down into several more focused questions that create a more descriptive picture of the problem. These questions can include:

- Who, or what, is reporting the problem?
- What are the error codes and messages?
- How does the system fail? For example, is it a loop, hang, crash, performance degradation, or incorrect result?

- What is the business impact of the problem?

Where does the problem occur?

Determining where the problem originates is not always easy, but it is one of the most important steps in resolving a problem. The following questions can help you to focus on where the problem occurs in order to isolate the problem area.

- Is the problem specific to one platform or operating system, or is it common across multiple platforms or operating systems? For example, is the service requester on a different platform?
- Is the current environment and configuration supported? For example, are you using one of the supported interfaces when trying to access the service flow?
- Were there any problems with the service flow when it was modeled? Errors that occur in the service flow can manifest as problems at run time.
- Is the problem specific to one service flow or server adapter?

Remember that, even though one area might report the problem, the problem does not necessarily originate there. Part of identifying where a problem originates is understanding the environment in which it exists. Take some time to completely describe the problem environment, including the operating system, its version, all corresponding software and versions, and hardware information. Confirm that you are running in an environment that is a supported configuration; many problems can be traced back to incompatible levels of software that are not intended to run together or have not been fully tested together.

When does the problem occur?

Develop a detailed timeline of events leading up to a failure, especially for those cases that are one-time occurrences. You can most easily do this by working backward: Start at the time an error was reported (as precisely as possible, even down to the millisecond), and work backward through the available logs and information. Typically, look only as far as the first suspicious event that you find in a diagnostic log; however, finding that event is not always easy and takes practice. Knowing when to stop looking is especially difficult when multiple layers of technology are involved, and when each has its own diagnostic information.

To develop a detailed timeline of events, try to answer these questions:

- Does the problem happen only at a certain time?
- How often does the problem happen?
- What sequence of events leads up to the time that the problem is reported?
- Does the problem happen after an environment change, such as upgrading or installing software or hardware?

Responding to questions like these can help to provide you with a frame of reference in which to investigate the problem.

Under which conditions does the problem occur?

Knowing what other systems and applications are running at the time that a problem occurs is an important part of troubleshooting. These and other questions about your environment can help you to identify the root cause of the problem:

- Does the problem always occur when the same task is being performed?
- Does a certain sequence of events lead to the problem?
- Do any other applications fail at the same time?
- Which version of the tooling did you use to model your service flows? Have you regenerated the service flows on a later version?

Answering these types of questions can help you explain the environment in which the problem occurs and correlate any dependencies. Remember, just because multiple problems might have occurred around the same time, the problems are not necessarily related.

Can the problem be reproduced?

From a troubleshooting standpoint, the "ideal" problem is one that can be reproduced. Typically with problems that can be reproduced, you have a larger set of tools or procedures at your disposal to help you investigate. Consequently, problems that you can reproduce are often easier to debug and solve. It is recommended that if the problem is of significant business impact, recreate the problem in a test or development environment, which typically offers you more flexibility and control during your investigation.

- Can the problem be recreated on a test machine?
- Are multiple users or applications encountering the same type of problem?
- Can the problem be recreated by running a single command, a set of commands, or a particular application, or a stand-alone application?

About fixes and updates

If you encounter a problem, first check the list of APARs to see whether IBM has already published a fix that resolves your problem. Individual fixes are published as often as necessary to resolve defects in CICS Service Flow Runtime.

APARs are listed on the IBM support site for CICS SFR. Selecting this link displays all the APARs that are related to CICS SFR. You can add additional search terms that relate to your problem if you would like to refine the query further. You can also change the ordering of the search results, for example, to show the most recent APARs first.

If you select an APAR from the search results, check the details to see if it matches your problem. If a fix is available, it is displayed at the top of the page. Most APARs that have fixes are optional. However, if an APAR has the highly pervasive (HIPER) field listed as `YesHIPER` in the APAR information section at the bottom of the page, you must apply the fix.

The fix information itself is underneath the APAR information section. It contains the component name and ID for the CICS Service Flow Feature and the component level. If you select the PTF link for the component, you are directed to the zSeries related fixes Web site where you need to sign in to order the fix. A number of different services are available from this Web site, so you can select the most suitable service for ordering the fix.

Alternatively, you can access RETAIN and search on the feature name or number to list all the fixes (PTFs) that are available. You can then order the ones that you want from that list. This list is called a PSP Bucket.

If you think your problem is related to service flow project tools, check the IBM support site for WD/Z for interim fixes and updates that might solve your problem.

Troubleshooting aids

To troubleshoot problems with CICS Service Flow Runtime, you can use the following tools and utilities.

Vector logging

Vector logging is a diagnostic tool that allows you to record the flow of data between CICS applications and a virtual terminal. You can use it in a development environment to assist in the deployment and running of a service flow that includes Link3270 server adapters.

You can set vector logging as an option when a flow is modeled in service flow project tools or you can set it with the flow management transaction CMAN. When the service flow is deployed, the service flow properties file indicates that vector logging will be performed for that specific Link3270 server adapter.

Alternatively, if you want to turn vector logging on for a Link3270 server adapter that is already deployed, use the CMAN transaction.

You can select two levels of granularity for vector logging:

Full vector logging
This level of logging records the header structures and the inbound and outbound vectors including any vector data. The vector data can be an inbound or outbound application data structure (ADS), text, or 3270 datastream.

Vector logging trace
This level of logging records the header structures and the inbound and outbound vectors. It does not record any vector data.

When you switch on vector logging, the Link3270 server adapter writes to the active vector log file. Two files, DFHMALVA and DFHMALVB, store vector logging information. One of these files is always active and the other is either empty or contains old data. The Link3270 server adapter starts by writing vector logging data to DFHMALVA. When the file is full, the Link3270 server adapter swaps to continue writing information to DFHMALVB. When DFHMALVB is full, the server adapter moves back to writing to DFHMALVA, deleting the old content.

By using two files, you can analyze the complete vector logging data for a server adapter. A server adapter is not likely to write enough data to fill both files, but if it does happen, the most recent data is always available.

You can access the contents of the vector log file by dumping it using the provided sample JCL job DFHMAMVD. This job runs the dump utility DFHMAVUP, which formats the file so that you can read it to help with problem determination. The dump utility displays the records in chronological order.

CICS dump and traces

CICS dumps and traces are an important source of detailed information about problems.

Whether they are the result of an abend or a user request, dumps allow you to see a snapshot of what was happening in CICS at the moment the dump is taken. However, because they do provide a snapshot, you might need to use them with other sources of information relating to a longer period of time, such as logs, traces, and statistics.

For information about the dump formatting keywords used to extract BTS information from a CICS system dump, see Using dumps in problem determination.

CICS also provides tracing that enables you to trace transactions through the CICS components and service flow programs.

Troubleshooting checklist

The following checklist can help you to identify the source of the problem that is occurring in the runtime environment.

About this task

For recommended responses to specific errors, see the applicable sections of “Messages and codes” on page 118.

Procedure

1. Do you have the latest APARs applied?

   IBM might have already published a fix for your problem. See “About fixes and updates” on page 91 for details on how to check for the latest available fixes.
2. Does the problem occur when you are applying an APAR?

   a) Ensure that you are following the correct process to apply the APAR.

This is described in “Applying APARs” on page 149.

b) If you are following the correct process and still have difficulties, contact IBM Software Support.

3. Are you having problems when running the postinstallation jobs?

   For example, you receive a return code other than 0 when a job completes.

   For information on diagnosing errors with the post-installation jobs, read “Troubleshooting postinstallation errors” on page 94.

4. Does the problem occur when you are deploying a service flow?

   For example, the service flow installs in an unusable state.

   a) Check that the PROCESSTYPE resource for the service flow is installed in the CICS region.

      The name of the resource is the same as the request name of the service flow.

   b) If the PROCESSTYPE resource has been discarded, reinstall it.

   c) Try to install the service flow again using the CMAN transaction.

5. Does the problem occur when a service requester invokes a deployed service flow?

   When a problem occurs in the runtime environment, errors are written to the CMAC transient data queue. The message contains the details of the error.

   a) Analyze the message to determine if a server adapter is causing the problem.

      The `Error` field contains the message ID. Read the “Messages and codes” on page 118 to find the type of error that has occurred. Also take note of the `Program` field, because it indicates the program in which the error occurred which might be the name of a server adapter.

   b) If you receive a DFHMA06021E message, check the request name to ensure that it is the name of deployed service flow.

      Also check that the service flow properties file for the service flow is in the deployment directory.

   - Is the problem related to a Link3270 adapter? See “Troubleshooting Link3270 server adapters” on page 96.
   - Is the problem related to the Web services server adapter? See “Troubleshooting the web service server adapter” on page 98.
   - Is the problem related to a FEPI server adapter? See “Troubleshooting FEPI server adapters” on page 95.

6. Optional: Dump and analyze the CICS BTS audit trail if you have configured it for use.

   See “Using a BTS audit trail for problem determination” on page 99 for more information.

7. Debug your applications.

   See “Debugging your application” on page 101 for more information.

8. Dump and analyze any CICS trace and dump information available.

   See “Using CICS trace for problem determination” on page 100 and “CICS dump and traces” on page 92 for more information.

9. Use additional CICS-supplied transactions.

   See “Using CBAM for problem determination” on page 101 for more information.

Results

If this checklist does not guide you to a resolution, contact IBM to report a problem. Before you contact IBM, you might have to collect additional diagnostic data. This data is required when reporting a problem to IBM and can increase the problem resolution time. For details on the information to collect and how to send it to IBM, see Collecting CICS troubleshooting data (CICS MustGather) for IBM Support.

Troubleshooting postinstallation errors

The postinstallation jobs are DFHMAINJ and DFHMASET and both can return response codes other than 0.

Procedure

- If you are running the DFHMAINJ sample job with validation enabled and it fails with a return code other than 0:
  a) Check the job output for any messages that are prefixed with DFHMAI.
     They indicate which parameter values have caused a problem.
  b) Fix the problem and rerun the job.
- If you are running DFHMAINJ without validation enabled and the job fails with a return code other than 0:
  a) Check the job output to see which error messages were issued.
     They will give you an indication of what caused the job to fail. Update the values in DFHMAINJ.
  b) Uncomment the section that is marked in DFHMAINJ to delete the customized libraries when the job is next invoked.

```
//*-------------------------------------------------------
//* To rereun DFHMAINJ, uncomment this step to delete the -
//* data sets before recreating them. -
//* -
//* It is vitally important that an empty SCIZSAMP -
//* data set exists before executing the REXX step. -
//* @BA32131A -
//*-------------------------------------------------------
//*SYSD2 EXEC PGM=IKJEFT01
//*SYSUDUMP DD SYSOUT=*
//*SYSTSPRT DD SYSOUT=*
//*SYSPRINT DD SYSOUT=*
//*SYSTSIN DD *
//* DEL 'hlqual.SCIZSAMP'
//* DEL 'hlqual.SCIZLOAD'
//* DEL 'hlqual.SCIZMAC'
```

  c) Enable validation for the job by changing `novalidate` to `validate` on the DFHMAINR invocation statement in the //REXX step.
  d) Rerun DFHMAINJ.
     If you still see a return code other than 0, check the job output for DFHMAI messages. They will tell you why the job is invalid. Fix the job as appropriate.
  e) Comment out the delete statements in DFHMAINJ and then rerun it.
     When you run the job with validation enabled, the libraries are not customized when there is an error so you do not hvae to include the deletion statements.
- If you are running the DFHMASET job, a return code other than 0 might be returned for some of the steps.
  - If you see a return code of 4, accompanied by error message IGYDS0001 when compiling DFHMADCD, DFHMADCI, and DFHMAVUP, this error message is acceptable for these steps.
  - If you have specified compiler options OPTIMIZE(STD) or OPTIMIZE(FULL), a return code of 4, accompanied by error message IGYOP3091-W, is also acceptable for these steps.
  - If you see a return code that is not 0 or 4, contact IBM Software Support, providing a copy of the DFHMASET job that you used and the CICS job log.

Troubleshooting FEPI server adapters

If you are experiencing a problem with a FEPI server adapter, read through the following information to find out how to diagnose the problem.

About this task

If you receive an error message in the range DFHMA04001E to DFHMA04012E, or unexpected or incorrect data is being returned when you invoke the service flow, you must capture diagnostic information using the provided utilities and troubleshooting aids. FEPI adapter error messages contain the failed program name, transaction, target program name, target APPLID, conversation ID, CICS response codes, External Security Manager (ESM) return codes, and other relevant information:

```
Error detail: FEPI
CICS Resp: 00000005 CICS Resp2: 00000006 Convid: 60260459 44397251 Transid: TRI*
Pool name: PoolNam* Target name: FEPITAR* Target Applid: Japplid* Node name: JNode67*
Node owner: FNodeOwner123456789+123* Propertyset: FPSet67* ESM Resp: 00000005 ESM Reason: 00000006
```

Use this information to assist in problem diagnosis.

Follow the steps below to collect the required information and analyze it to find the cause of the error.

Procedure

- You receive a DFHMA04002E message with a CICS RESP2 code of 30.
  a) Check the POOL and TARGET names in the DFHMA04002E message and the generation properties of the service flow in service flow project tools to ensure they are valid for the FEPI environment in the CICS region.
  b) Use **CEMT INQUIRE FEPOOL** to determine the pool names defined in the CICS region and **CEMT INQUIRE FETARGET** to determine the target name and APPLID of the CICS region.
  c) If the generation properties of the service flow are incorrect, correct the service flow and redeploy it to CICS.
  d) If the generation properties of the service flow are correct, ensure that the **FEPI** system initialization parameter is set to YES in the CICS region.

     If FEPI is not specified in the CICS region:

     – Set the **FEPI** system initialization parameter to YES.
     – Customize and compile a FEPI sample setup program in the CICS region.
     – Run the CZXS transaction in the CICS region and rerun the FEPI flow.
     – Rerun the FEPI service flow.
  e) If FEPI is correctly installed in the CICS region, check that the property set in the FEPI sample setup program does not contain transactions that are not supported by CICS SFR:

     – In the FEPI setup program DFH0*ZXS, check the property set references by the POOL specified in the service flow properties file.
     – Ensure that none of the FEPI sample program transactions are included in the definition of this property set. These transactions are: CZUC, CZUU, CSZX, and CZUX.
     – When you delete the transactions from the property set definition, ensure that you overwrite them with four spaces between the quotes.
     – Recompile the program and rerun the FEPI service flow.
  f) If there are no FEPI transactions defined in the property set, check that there are enough NODE definitions to handle concurrent service flow requests:

     – Check the FEPI setup program DFH0*ZXS. Ensure that the number of NODEs defined in the specified POOL for the generation properties of the service flow matches the number of concurrent service flow requests that you expect to run against the TARGET. You can find the POOL and TARGET in the DFHMA04002E message.

  - Use **CEMT INQUIRE FENODE** to list the currently defined nodes.
  - If there are not enough nodes defined, you can use another POOL with a larger number of defined nodes. For guidance on determining the number of required nodes, see Configuring FEPI.
- The service flow returns the incorrect screen, the service flow hangs, or you receive a DFHMA04004E with RESP2 value of 213.

  This error could be caused by a problem with the number of screens that have been modeled or the Skip Receive count.

  a) To check that a RECEIVE DATASTREAM has not been issued for more screens than exist in the application, ensure that the application has not changed since the service flow was modeled.

  You can check the sequence of screens by running and formatting a CICS auxiliary trace with the SZ component. The AP1240 and AP1244 trace entries record the SENDs and RECEIVEs.

  b) If the application performs a sequence of SENDs with no intervening RECEIVE, check that the Skip receives fields on the FEPI generation properties panel in service flow project tools correctly specifies the number of screens to be bypassed before a RECEIVE is completed.

  The Skip Receives field is set for each terminal operation in the service flow, so ensure that the count is right for each operation.

  - If the Skip Receives field is set to less than the number of screens to be bypassed, the service flow is likely to return the incorrect screen and the service flow might fail.
  - If the Skip Receives field is set to a greater value than the number of actual screens to be bypassed, the service flow is likely to hang. This error is caused by the service flow trying to run more RECEIVEs than the number of screens that can be received.
- If the service flow fails or returns the wrong screen, a possible cause is that the FEPI service flow was modeled with the CICS CESN signon screen as the first screen, but the CICS region has since been started without GMTRAN=CESN specified.

  a) Run a CICS auxiliary trace with the SZ component.

  b) Format the trace and check the sequence of the AP 1240 SZATR entries for the RECEIVE and the associated AP 1244 SZATR EVENT records, specifically for the screen being received at the time of the failure.

  c) Restart the CICS region with GMTRAN=CESN in the startup JCL.

  d) Rerun the FEPI service flow
- If a service flow fails or returns the wrong screen, the screens of the application on which the FEPI service flow was modeled might have changed.

  a) Check the screen recognition criteria that was used to capture the screens during modeling and compare with the application screens.

  Check that a field used within a field pattern descriptor does not contain data that might vary or that contains data that has changed since the service flow was modeled; for example, timestamps or dates.

  b) If the flow modeler selected to recognize screens by the number of fields, number of input fields, or the checksum, ensure that the number and type of fields on the application screens remain the same.

  c) If there are any differences, use the service flow project tools to update the service flow with the new screens and redeploy the service flow to CICS.

Troubleshooting Link3270 server adapters

If you are experiencing a problem with a Link3270 server adapter, read through the following information to find out how to diagnose the problem.

About this task

The Link3270 server adapter enables a service requester to access 3270 application programs that are running in CICS using the CICS Link3270 bridge mechanism. If you receive an error message in the range

DFHMA07001E to DFHMA07999E, or unexpected or incorrect data is being returned when you invoke the service flow, you must capture diagnostic information using the provided utilities and troubleshooting aids. Follow the steps below to collect the necessary information and analyze it to find the cause of the error.

Procedure

1. Check the CICS job log for any recent error messages.

   For example, if you see a DFHBR0501 error message, the Link3270 server adapter has failed because the Link3270 bridge file DFHBRNSF is not defined in your CICS region. This file is required to run any Link3270 server adapter.
2. If there are any DFHMA-prefixed error messages, check the error information provided with the message to give you an indicator of what went wrong.
   a) Look up the values for the CICS RESP and RESP2 fields in CICS Application development reference.

      The RESP and RESP2 codes will give you a good idea of why there was a failure.
   b) If the error occurred in a Link3270 server adapter, the end of the error message displays the inbound and outbound Link3270 bridge message field values.
3. If you receive a DFHMA02002E error message for a complex Link3270 service flow, ensure that the SERVICE NAME field in the generation properties of service flow project tools is set to the same value for each Link3270 bridge node.

   CICS SFR uses this field to carry status information about the server adapters. If the field is omitted, it can cause this error.
4. If you need to capture additional information, enable vector logging for your server adapter using the CMAN transaction.
   a) Enter `CMAN` in the display.

      Browse the list to find the service flow that is causing the error.
   b) Press 1 to view the service flow details.
   c) Enter 2 next to the appropriate Link3270 server adapter to switch on vector logging.
5. Invoke the service flow to capture the flow of data through the Link3270 server adapter.

   CICS SFR updates the vector log file during request processing.
6. Dump the vector log file using the provided sample JCL.
   a) Check that the sample module DFHMAVUP is located in the *hlq*.SCIZSAMP library and has been compiled.
   b) Run the sample JCL job DFHMAMVD.

      This runs the vector log dump utility DFHMAVUP. See “Link3270 Vector Log file dump JCL, DFHMAMVD” on page 154 to view the sample JCL.
7. Analyze the vector log file dump.

   See “Analyzing the vector log file dump” on page 97 for what information to look for in the dump.

Analyzing the vector log file dump

The vector log file dump displays the flow of data between CICS applications and a virtual terminal.

About this task

The vector file dump can contain the header structures, the inbound and outbound vectors, and optionally the vector data.

Procedure

1. Review the header of the vector file dump to find the context of where the error occurred.

For example, it tells you the request name that was being processed when the error occurred, the name of the Link3270 server adapter program, the BTS process type and name that the server adapter was running under. This header is then followed by the vectors and flow of data that occurred.

2. Check each inbound and outbound vector message structure (BRIV).

   Each one is preceded by the Link3270 bridge header structure, BRIH.

   a) Check the Return code fields to see if the value is zero or another number. If it is not zero, the Link3270 bridge mechanism has reported an error.

      This error is the result of an earlier inbound ALLOCATE facility message, DELETE facility message, or inbound vector message (BRIV) sent from CICS Service Flow Runtime to the Link3270 bridge mechanism, (DFHL3270).

   b) Check the completion and reason codes.

      If these are not 0, look up the codes in Developing for external interfaces to find an indication of what error occurred.

Example

For an annotated example of the vector log file, see "Vector file dump" on page 154.

Troubleshooting the web service server adapter

If you are experiencing a problem with the web service server adapter, read through the following information to find out how to diagnose the problem.

About this task

The web service server adapter DFHMASWS enables a service flow to send a web service request to a service provider using the existing web services support in CICS. If the error message DFHMA08112E is in the error file dump or a SOAP fault message, it indicates that an error occurred in the runtime environment when it tried to issue the web service request.

Procedure

1. Check the CICS job log for any DFHPI-prefixed error messages.

   A DFHPI-prefixed message can indicate that the web services support in the CICS region has an underlying problem.

   If you have DFHPI messages, follow the guidance in the message details to fix the problem. You can also check Troubleshooting SOAP web services.

2. Analyze the details of the error message to see if it indicates what the problem is.

   a) Look for the error message DFHMA08112E.

      The details of the error entry are below the message.

      Here is an example of what you might see:

```
--------------------------------------------------------------------------------------------
-----------------
Processed:  Date:  04/27/06     Time:  13:48:04:       PutApplid:
PutTranid:
    Error:  DFHMA08112E    Normal
processing


   Userid:  CICSUSER     Applid:               Tranid:  CMAO     Eibtaskn:   0000093
AbsTime:  003355134483920
  Request:  SAMPCARN       Mode:  Sync         Program:  DFHMASWS     Type:
System
 Activity:  PlaceOrder                      Node
Name:
    Event:  DFHINITIAL                    Event type:  System       Step:
MAIN
 Proctype:  DFHMAINA                         Process:
```

```
003355134483840T160
Failed Processtype:                    Failed
Process:
 ReplyToQ:
ReplyToQMgr:
 MQ MsgId:                                MQ
CorrelId:


Error detail:  Web Service
request
 Web Service Resource Name:  testPlaceOrder                     CICS Resp:  00000016
CICS Resp2:  00000004
     Web Service Operation:
DFH0XCMNOperation
Overriding Web Service URI:
12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
------------------
```

In the example, the value `Program: DFHMASWS` indicates that the error occurred in the web services server adapter.

b) Check the meaning of the CICS RESP2 code for the INVOKE WEBSERVICE command.

In the previous example, CICS returned a RESP code of 16 and a RESP2 code of 4, which indicate that the URI is not valid.

3. Fix the problem and invoke the service flow again to ensure it works correctly.

In the previous example, you must change the URI that is passed to the web service server adapter. Depending on the nature of the problem, you might have to redeploy your service flow into CICS. For details on how to do this, see "Updating an existing service flow" on page 55.

Using a BTS audit trail for problem determination

You might want to create an audit trail for the BTS processes and activities that are used by a service flow to run in your CICS systems. An audit trail can assist in problem determination.

About this task

Audit log records are written to an MVS™ logstream by the CICS log manager. You can read the records offline using the CICS audit trail utility program DFHATUP.

Procedure

1. Use the AUDITLOG and AUDITLEVEL attributes of the PROCESSTYPE resource for the service flow to control the audit logging that takes place and where the logs are stored.

Use caution when creating audit trails in production because they can significantly affect system performance.

a) Use the **INQUIRE PROCESSTYPE** command to check that an audit log has been defined.

If this attribute is not defined for the resource, audit logging cannot take place. If an audit log is not defined, delete the existing resource and create a new resource definition with the AUDITLOG attribute specified.

b) To enable or disable audit logging, or to change the type of audit logging that is taking place, use the **SET PROCESSTYPE** command.

The audit levels that you can select are:

- ACTIVITY
- FULL
- OFF
- PROCESS

Changing the audit logging with this command has no effect on any existing processes that are running in the CICS system for that process type. Only new processes of that process type write audit records to the audit log.

2. To read the records in the audit log, use the sample job DFHMABAP, which is located in the SCIZSAMP samples library.

   This job runs the DFHATUP utility. See "Audit file dump JCL, DFHMABAP" on page 153 to view the sample JCL.

What to do next

For detailed information about audit levels and configuration, see CICS Business Transaction Services.

Using CICS trace for problem determination

CICS provides tracing that enables you to trace transactions through the CICS components and through your own programs. You can either define the tracing levels at system initialization or use a CICS-supplied transaction to define tracing when CICS is running.

About this task

To define tracing when CICS is running, use the CETR transaction.

Procedure

1. Use the CETR transaction to run level 1 auxiliary tracing for the BTS domains.

   BTS consists of three CICS domains: the Business application manager domain, the Event manager domain, and the Scheduler services domain. You can trace what is happening in BTS using the component codes for these domains to specify the level of standard and special tracing for BTS. The component codes are as follows:

Domain name	CICS component code
Business application manager	BA
Event manager	EM
Scheduler services	SH

   This tracing provides you with information about the BTS processes and activities that are running in the CICS region.

2. Use the CETR transaction to run level 1 auxiliary tracing for the following domains:

   - AP - application programming domain
   - PG - program manager domain

   This tracing provides you with information about the programs that are running and the contents of the BTS data-containers. CICS SFR also performs user tracing to trace the entry and exit of the modules that are called during request processing. See "Trace points" on page 101.

3. If you are having problems with Web services in particular, use the CETR transaction to run level 1 auxiliary tracing for the PI domain.

   CETR traces what is happening in the pipeline when a service requester invokes a service flow from a Web service, or if a service flow makes an outbound Web service request.

Debugging your application

The CICS execution diagnostic facility (EDF) helps you when using the EXEC CICS interface to step through the EXEC CICS commands of an application program.

About this task

Use EDF to debug your server adapters; adapters are CICS applications that use the CICS BTS API. The sample compile PROC has the NOEDF option specified.

Because the service flows do not run attached to a terminal, you must use the CEDX transaction specifying the transaction ID of the flow navigator or server adapter that you want to debug.

See Debugging applications. In addition, it can be useful to use a third-party debugging product that can provide a COBOL verb-at-a-time capability.

Using CBAM for problem determination

You might also find it useful and even necessary to inquire on and control CICS BTS resources.

About this task

Use the CEMT transaction to inquire on a BTS PROCESSTYPE or task and use the BTS browser transaction CBAM to assist in problem diagnosis.

You can use CBAM to browse the CICS BTS objects (process types, processes, activities, containers, events, and timers) known to a particular region. For introductory and guidance information about the CICS main terminal transaction, CEMT, see CEMT - main terminal. For information on the CBAM transaction and on the useful CEMT commands in a BTS environment, see Administering BTS.

Trace points

When you switch on auxiliary tracing to help you diagnose problems, CICS SFR performs user tracing.

Most of the trace points show the entry and exit of the modules that are invoked during request processing rather than tracing data, although the contents of containers are also traced for request and response messages. Traces are also produced when an error occurs.

The following trace points are provided and each trace point has a number of trace resource IDs associated with it.

Trace point AP0065

Call and entry tracing for CICS SFR modules.

Module	Trace Resource ID	Lvl	Type	Data
DFHMADPL	MDPL01EN	AP 1	Entry	Program data
	MSDP01EN	AP 1	Entry	Program link adapter
	LDPL01CL	AP 1	Call	Link3270 adapter call
	QDPL01CL	AP 1	Call	Queue server adapter **MQPUT** call
	XDPL01CL	AP 1	Call	Called program ID
	XDPL02CL	AP 1	Call	Called program ID
	XDPL03CL	AP 1	Call	Called program ID
	XDPL04CL	AP 1	Call	Called program ID
DFHMAERH	MERH01EN	AP 1	Entry	Start of DFHMAERH processing

Module	Trace Resource ID	Lvl	Type	Data
DFHMAINS	MINS01EN	AP 1	Entry	Start of DFHMAINS processing
DFHMALFC	MLFC01EN	AP 1	Entry	Link3270 state cleanup file
DFHMALFD	MLFD01EN	AP 1	Entry	Link3270 facility deallocate
	LLFD01CL	AP 1	Call	Link3270 adapter call
DFHMALFS	MLFS01EN	AP 1	Entry	Link3270 facility state management
DFHMALIN	MLIN01EN	AP 1	Entry	Link3270 AOR routing
DFHMALNM	MLNM01EN	AP 1	Entry	Link3270 navigator
DFHMALSC	MLSC01EN	AP 1	Entry	Link3270 facility state cleanup (TSQ)
DFHMALTS	MLTS01EN	AP 1	Entry	Link3270 facility state management program (TSQ)
DFHMAMAN	MMAN01EN	AP 1	Entry	CMAN transaction entry
DFHMAMGR	MMGR01EN	AP 1	Entry	Navigation manager
	LMGR01CL	AP 1	Call	Link3270 adapter call
	XMGR01CL	AP 1	Call	Called program ID
	XMGR02CL	AP 1	Call	Called program ID
DFHMASCQ	MSCQ01EN	AP 1	Entry	Queue server adapter
DFHMASDP	DSDP01CL	AP 1	Call	Program link adapter LINK call
	DSDP02CL	AP 1	Call	Program link adapter LINK call
	DSDP03CL	AP 1	Call	Program link adapter LINK call
	DSDP04CL	AP 1	Call	Program link adapter LINK call
	DSDP05CL	AP 1	Call	Program link adapter LINK call
	DSDP06CL	AP 1	Call	Program link adapter LINK call
	DSDP07CL	AP 1	Call	Program link adapter LINK call
	DSDP08CL	AP 1	Call	Program link adapter LINK call
	DSDP09CL	AP 1	Call	Channel name and SYSID for program link
	DSDP10CL	AP 1	Call	Channel name, SYSID, and TRANSID for program link
	DSDP11CL	AP 1	Call	Channel name, SYSID, and SYNCONRETURN for program link
	DSDP12CL	AP 1	Call	Channel name, SYSID, TRANSID, and SYNCONRETURN for program link
	DSDP13CL	AP 1	Call	Channel name, SYNCONRETURN, and TRANSID
	DSDP14CL	AP 1	Call	Channel name and TRANSID
	DSDP15CL	AP 1	Call	Channel and SYNCONRETURN
	DSDP16CL	AP 1	Call	Channel name

Module	Trace Resource ID	Lvl	Type	Data
DFHMASWS	MSWS01EN	AP 1	Entry	Web services adapter
	WSWS01CL	AP 1	Call	Web service adapter before INVOKE WEBSERVICE with no overriding URI
	WSWS02CL	AP 1	Call	Web service adapter before INVOKE WEBSERVICE with overriding URI
DFHMAVCL	MVCL01EN	AP 1	Entry	Link3270 vector logging
DFHMAVCP	MVCPALEN	AP 1	Entry	Link3270 vector ALLOCATE processing
	MVCPDEEN	AP 1	Entry	Link3270 vector DELETE processing
	MVCPPREN	AP 1	Entry	Link3270 vector PROCESS processing with up to 4000 bytes of data
	LVCP01CL	AP 1	Call	Link3270 adapter COMMAREA with up to 4000 bytes of data
DFHMAF34	FF3401CC	AP 1	Call	FEPI adapter CONVERSE call
	FF3401CR	AP 1	Call	FEPI adapter RECEIVE call
	FF3401CS	AP 1	Call	FEPI adapter SEND call

Trace point AP0066

Return and exit tracing for CICS SFR modules.

Module	Trace Resource ID	Lvl	Type	Data
DFHMADPL	LDPL01RT	AP 1	Return	Link3270 adapter return
	MDPL01EX	AP 1	Exit	Module successful exit
	MDPL02EX	AP 1	Exit	Module error exit
	MSDP01EX	AP 1	Exit	Program link adapter exit
	QDPL01RP	AP 1	Return	Queue server adapter **MQPUT** return
	XDPL01RT	AP 1	Return	Called program ID return
	XDPL02RT	AP 1	Return	Called program ID return
	XDPL03RT	AP 1	Return	Called program ID return
	XDPL04RT	AP 1	Return	Called program ID return
DFHMAERH	MERH01EX	AP 1	Exit	DFHMAERH processing ended normally
DFHMAINS	MINS01EX	AP 1	Exit	DFHMAINS processing ended normally
	MINS02EX	AP 1	Exit	DFHMAINS processing encountered an error
DFHMALFC	MLFC01EX	AP 1	Exit	LINK3270 state cleanup file exit

Module	Trace Resource ID	Lvl	Type	Data
DFHMALFD	LLFD01RT	AP 1	Return	Link3270 adapter return
	MLFD01EX	AP 1	Exit	Link3270 facility deallocate exit
DFHMALFS	MLFS01EX	AP 1	Exit	Link3270 facility state management exit
DFHMALIN	MLIN01EX	AP 1	Exit	Link3270 AOR routing exit
	MLIN02EX	AP 1	Exit	Initiate exit with COMMAREA
	MLIN03EX	AP 1	Exit	Initiate exit without COMMAREA
DFHMALNM	MLNM01EX	AP 1	Exit	Link3270 navigator exit
DFHMALSC	MLSC01EX	AP 1	Exit	Link3270 facility state cleanup (TSQ) exit
DFHMALTS	MLTS01EX	AP 1	Exit	Link3270 facility state management program (TSQ) exit
DFHMAMAN	MMAN01EX	AP 1	Exit	CMAN transaction exit
DFHMAMGR	LMGR01RT	AP 1	Return	Link3270 adapter return
	MMGR01EX	AP 1	Exit	Navigation manager completed
	MMGR02EX	AP 1	Exit	Navigation manager did not complete
	MMGR03EX	AP 1	Exit	Navigation manager normal asynchronous return
	MMGR04EX	AP 1	Exit	Navigation manager error return
	XMGR01RT	AP 1	Return	Called program ID return
	XMGR02RT	AP 1	Return	Called program ID return
DFHMASCQ	MSCQ01EX	AP 1	Exit	Queue server adapter completed
DFHMASDP	DSDP01RT	AP 1	Return	Program link server adapter return
	DSDP02RT	AP 1	Return	Program link server adapter return
	DSDP03RT	AP 1	Return	Program link server adapter return
	DSDP04RT	AP 1	Return	Program link server adapter return
	DSDP05RT	AP 1	Return	Program link server adapter return
	DSDP06RT	AP 1	Return	Program link server adapter return
	DSDP07RT	AP 1	Return	Program link server adapter return
	DSDP08RT	AP 1	Return	Program link server adapter return
	DSDP09RT	AP 1	Return	Program link server adapter return

Module	Trace Resource ID	Lvl	Type	Data
	DSDP10RT	AP 1	Return	Program link server adapter return
	DSDP11RT	AP 1	Return	Program link server adapter return
	DSDP12RT	AP 1	Return	Program link server adapter return
	DSDP13RT	AP 1	Return	Program link server adapter return
	DSDP14RT	AP 1	Return	Program link server adapter return
	DSDP15RT	AP 1	Return	Program link server adapter return
	DSDP16RT	AP 1	Return	Program link server adapter return
DFHMASWS	MSWS01EX	AP 1	Exit	Web services adapter
	WSWS01RT	AP 1	Return	Web service adapter after INVOKE WEBSERVICE with no overriding URI
	WSWS02RT	AP 1	Return	Web service adapter after INVOKE WEBSERVICE with overriding URI
DFHMAVCL	MVCL01EX	AP 1	Exit	Link3270 vector logging exit
DFHMAVCP	LVCP01RT	AP 1	Return	Link3270 adapter COMMAREA, up to 4000 bytes
	MVCPALEX	AP 1	Exit	Link3270 vector ALLOCATE processing
	MVCPDEEX	AP 1	Exit	Link3270 vector DELETE processing
	MVCPPREX	AP 1	Exit	Link3270 vector PROCESS processing with up to 4000 bytes of data
	MVCP02EX	AP 1	Exit	Link3270 vector processing error exit
DFHMAF34	FF3401RC	AP 1	Return	FEPI adapter CONVERSE return
	FF3401RR	AP 1	Return	FEPI adapter RECEIVE return
	FF3401RS	AP 1	Return	FEPI adapter SEND return

Trace point AP0067

User trace point AP0067 traces data for the CICS SFR interface program and issues exception tracing for any error message that occurs.

Table 19. Tracing for module DFHMADPL

Trace Resource ID	Lvl	Type	Data
CDPL01RQ	AP 1	Data	DFHMAC-REQUESTV1 container contents of inbound request
CDPL01WD	AP 1	Data	DFHWS-DATA container contents of inbound request
CDPL01AP	AP 1	Data	DFHMAC-ALLPARMS container contents of inbound request
CDPL01UD	AP 1	Data	DFHMAC-USERDATA container contents of inbound request
CDPL01SP	AP 1	Data	DFHMAC-SYSPARMV1 container contents of inbound request
CDPL01LK	AP 1	Data	DFHMAC-LNK3270V1 container contents of inbound request
CDPL01SL	AP 1	Data	DFHWS-SOAPLEVEL container contents of inbound request
CDPL02WD	AP 1	Data	DFHWS-DATA response container contents
CDPL02AP	AP 1	Data	DFHMAC-ALLPARMS container contents of outbound response
CDPL02UD	AP 1	Data	DFHMAC-USERDATA container contents of outbound response
CDPL02ER	AP 1	Data	DFHMAC-ERROR container contents

Trace resource IDs that are produced when an error occurs are prefixed with SFR. The rest of the resource ID matches the number of the error message that is issued. For example, if message DFHMA01001E is issued, the associated trace resource ID that appears in the trace is SFR01001.

Table 20. Tracing for error messages

Trace Resource ID	Lvl	Type	Data
SFR01001	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR01002	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN
SFR01003	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN
SFR01004	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN
SFR01005	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN
SFR01006	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR01007	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN
SFR01008	AP 1	Data	FILE-NAME FILE-CODE FILE-FUNCTION FILE-RESP FILE-RESP2 FILE-KEY FILE-KEY-LEN FILE-LEN
SFR01331	AP 1	Data	TS-DATA-LENGTH TS-DATA
SFR01332	AP 1	Data	TS-DATA-LENGTH TS-DATA
SFR01333	AP 1	Data	TS-DATA-LENGTH TS-DATA
SFR01334	AP 1	Data	TS-DATA-LENGTH TS-DATA
SFR01335	AP 1	Data	TS-DATA-LENGTH TS-DATA
SFR01336	AP 1	Data	TS-DATA-LENGTH TS-DATA
SFR01337	AP 1	Data	TS-DATA-LENGTH TS-DATA
SFR02001	AP 1	Data	ERR-REC-DC-RESP ERR-REC-DC-RESP2 ERR-REC-DC-NAME ERR-REC-DC-OWNER ERR-REC-DC-DATA ERR-REC-DC-LEN

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR02002	AP 1	Data	ERR-REC-DC-RESP ERR-REC-DC-RESP2 ERR-REC-DC-NAME ERR-REC-DC-OWNER ERR-REC-DC-DATA ERR-REC-DC-LEN
SFR02003	AP 1	Data	ERR-REC-DC-RESP ERR-REC-DC-RESP2 ERR-REC-DC-NAME ERR-REC-DC-OWNER ERR-REC-DC-DATA ERR-REC-DC-LEN
SFR02004	AP 1	Data	ERR-REC-DC-RESP ERR-REC-DC-RESP2 ERR-REC-DC-NAME ERR-REC-DC-OWNER ERR-REC-DC-DATA ERR-REC-DC-LEN
SFR03001	AP 1	Data	ERR-REC-DPL-RESP ERR-REC-DPL-RESP2 ERR-REC-DPL-PROGRAM ERR-REC-DPL-SYSID ERR-REC-DPL-TRANSID ERR-REC-DPL-LENGTH ERR-REC-DPL-DATALENGTH ERR-REC-DPL-DATA ERR-REC-DPL-SYNCONRETURN
SFR04001	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR04002	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR04003	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR04004	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR04005	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR04006	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR04007	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR04008	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR04009	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR04010	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR04011	AP 1	Data	ERR-REC-FEPI-RESP ERR-REC-FEPI-RESP2 ERR-REC-FEPI-TARGET ERR-REC-FEPI-TARGET-APPLID ERR-REC-FEPI-NODE ERR-REC-FEPI-POOL ERR-REC-FEPI-NODE-OWNER ERR-REC-FEPI-CONVID ERR-REC-FEPI-TRANSID ERR-REC-FEPI-ESM-RESP ERR-REC-FEPI-ESM-REASON ERR-REC-FEPI-PROPERTYSET
SFR05001	AP 1	Data	ERR-REC-MQ-OBJECTTYPE ERR-REC-MQ-OBJECTNAME ERR-REC-MQ-OBJECTQMGRNAME ERR-REC-MQ-ALTERNATEUSERID ERR-REC-MQ-RESOLVEDQNAME ERR-REC-MQ-RESOLVEDQMGRNAME ERR-REC-MQ-RESOLVEDQMGR ERR-REC-MQ-COMPCODE ERR-REC-MQ-REASON
SFR05002	AP 1	Data	ERR-REC-MQ-OBJECTTYPE ERR-REC-MQ-OBJECTNAME ERR-REC-MQ-OBJECTQMGRNAME ERR-REC-MQ-ALTERNATEUSERID ERR-REC-MQ-RESOLVEDQNAME ERR-REC-MQ-RESOLVEDQMGRNAME ERR-REC-MQ-RESOLVEDQMGR ERR-REC-MQ-COMPCODE ERR-REC-MQ-REASON

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR05004	AP 1	Data	ERR-REC-MQ-OBJECTTYPE ERR-REC-MQ-OBJECTNAME ERR-REC-MQ-OBJECTQMGRNAME ERR-REC-MQ-ALTERNATEUSERID ERR-REC-MQ-RESOLVEDQNAME ERR-REC-MQ-RESOLVEDQMGRNAME ERR-REC-MQ-RESOLVEDQMGR ERR-REC-MQ-COMPCODE ERR-REC-MQ-REASON
SFR05005	AP 1	Data	ERR-REC-MQ-OBJECTTYPE ERR-REC-MQ-OBJECTNAME ERR-REC-MQ-OBJECTQMGRNAME ERR-REC-MQ-ALTERNATEUSERID ERR-REC-MQ-RESOLVEDQNAME ERR-REC-MQ-RESOLVEDQMGRNAME ERR-REC-MQ-RESOLVEDQMGR ERR-REC-MQ-COMPCODE ERR-REC-MQ-REASON
SFR06001	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06002	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06003	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR06004	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06005	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06006	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06007	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06011	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR06017	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06018	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06021	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR06022	AP 1	Data	BTS-RESP BTS-RESP2 BTS-COMPSTATUS BTS-MODE BTS-SUSPSTATUS BTS-ABCODE BTS-PROGRAM BTS-ACTIVITY BTS-TRANSACTION
SFR07001	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07002	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR07010	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07011	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07012	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07013	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07014	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07015	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07016	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR07017	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR07901	AP 1	Data	EDC-LC-RESP EDC-LB-RESP2 EDC-LB-PROGRAM EDC-LB-LENGTH EDC-LB-DATA
SFR08001	AP 1	Data	CICS-RESP CICS-RESP2
SFR08002	AP 1	Data	CICS-RESP CICS-RESP2
SFR08006	AP 1	Data	CICS-RESP CICS-RESP2
SFR08008	AP 1	Data	BIDITRN-RESP BIDITRN-RESP2 BIDITRN-MODULE BIDITRN-DATA BIDITRN-DATALEN BIDITRN-INATTR BIDITRN-INATTRLEN BIDITRN-OUTATTR BIDITRN-OUTATTRLEN BIDITRN-CODEPAGE BIDITRN-CODEPAGELEN
SFR08010	AP 1	Data	
SFR08011	AP 1	Data	
SFR08101	AP 1	Data	CICS-RESP CICS-RESP2
SFR08102	AP 1	Data	CICS-RESP CICS-RESP2
SFR08103	AP 1	Data	CICS-RESP CICS-RESP2
SFR08104	AP 1	Data	CICS-RESP CICS-RESP2
SFR08106	AP 1	Data	CICS-RESP CICS-RESP2
SFR08107	AP 1	Data	CICS-RESP CICS-RESP2

Table 20. Tracing for error messages (continued)

Trace Resource ID	Lvl	Type	Data
SFR08108	AP 1	Data	CICS-RESP CICS-RESP2
SFR08109	AP 1	Data	CICS-RESP CICS-RESP2
SFR08110	AP 1	Data	CICS-RESP CICS-RESP2
SFR08111	AP 1	Data	CICS-RESP CICS-RESP2
SFR08112	AP 1	Data	
SFR08301	AP 1	Data	XML code XML data length XML data
SFR99999	AP 1	Data	Abend code ERR-REC-ABDUMP ERR-REC-ABPROGRAM ERR-REC-SYSID ERR-REC-ASSIGN-RESP ERR-REC-ASSIGN-RESP2

Trace point AP0068

User trace point AP0068 traces service flow events, enabling you to relate the processing and information provided by other trace points to the progress through the service flow.

These trace points are defined in the service flow using service flow project tools. For FEPI and LINK3270 server adapters, the module is the name of the service flow that is requesting to run the adapter. If the service flow has one or more subflows, the flow navigator program traces a SUBFLOW resource ID.

Table 21. Tracing for service flow events

Module	Trace resource ID	Lvl	Type
DFHMASDP	CHANNEL	AP 1	Data
DFHMASDP	COMMAREA	AP 1	Data
service_flow	FEPI	AP 1	Data
service_flow	LINK3270	AP 1	Data
flow_navigator	SUBFLOW	AP 1	Data
DFHMASWS	WEBSERV	AP 1	Data
DFHMASCQ	WMQ	AP 1	Data

Messages and codes

CICS Service Flow Runtime issues information, warning, and error messages during events such as installation, deployment, and service flow processing. Any message that is issued by CICS SFR is written

to the transient data queue CMAC. CMAC is an alias for CSMT, so that the CICS SFR messages appear alongside other CICS messages.

The following messages and codes can be issued by CICS SFR.

Format of messages

The information that is displayed in the text of each message has a standard set of fields, although not all of the fields are populated with data for certain messages. The messages are also grouped into types, and each type of message includes some additional information to help diagnose the cause of the error.

The standard set of fields appear in each message in the following format:

```
Processed: Date: ddmmyy Processed: Time: hhmmss PutApplid: applid PutTranid: transaction
    Error: message number Processing type

   Userid: username        Applid: applid Tranid: tranid   Eibtaskn: number AbsTime: time
  Request: request name      Mode: mode  Program: program name Type: program type
 Activity: activity name                              Node name: node name
    Event: event                                   Event type: event type   Step: program
step
 Proctype: process type                                  Process: process
Failed Processtype: failed process                Failed process: failed process
 ReplyToQ: WebSphere MQ reply queue                          ReplyToQMgr: WebSphere MQ reply
queue mgr
 MQ MsgId: WebSphere MQ message identifier                   MQ CorrelId: WebSphere MQ
correlation identifier

Error detail: error text
```

The values for these fields are described below:

Processed: Date
The date that the record was written to the CMAC transient data queue.

Processed: Time
The time that the record was written to the CMAC transient data queue.

PutApplid
The APPLID retrieved from the origin context in the WebSphere MQ Message Descriptor structure (MQMD), after a successful **MQGET** command. See the IBM MQ product documentation for more information.

PutTranid
The transaction ID retrieved from the original context in the WebSphere MQ Message Descriptor structure (MQMD), after a successful **MQGET** command.

Error
The error message that indicates the error that occurred. See “Messages and codes” on page 118 for a description of the error messages that can display in this field.

Processing type
The mode of request processing. The value is derived from a field in the request message header. See “Request message headers” on page 47 for information on the DFHMAH-UOWCONTROL field.

Valid values are as follows:

- Normal processing
- Compensation processing
- Cancel processing

Userid
The user ID under which the error was reported, and for which the request processing was occurring.

Applid
The APPLID indicating the CICS region and transaction, under which the failed application program was running, where the error occurred.

This value can be different from the PutApplid if an error occurred in a program link or queue server adapter that was not running in the same region as the controlling flow navigator; for example, because of workload balancing. The program link and queue server adapters return control to the flow navigator indicating the error. The flow navigator subsequently writes to the CMAC TDQ.

Tranid
The transaction ID indicating the CICS region and transaction, under which the failed application program was running, where the error occurred.

Eibtaskn
The task number of the failed transaction assigned by CICS. See CICS Application development reference for more information.

AbsTime
The time as reported by CICS when the error occurred. This time is retrieved with an **EXEC CICS ASKTIME** command. See CICS Application development reference for more information

Request
The name of the service flow request that was being processed when the failure occurred.

Mode
The mode of request processing that was in effect when the failure occurred. The type of request processing is saved in the service flow repository file, DFHMAASF, and is retrieved based on the request name. Valid values are as follows:

- Async
- Link
- Sync
- Sync RB

Program
The program name that reported the error or where the failure occurred. In the case of a program link server adapter, it could be the target program name. This is true when the error is an abend in the target program.

Type
The type of program indicated by the program name. Valid values are as follows:

- Navigator
- DPL
- MQPUT
- FEPI
- MQGET
- LINK3270
- System

A value of `system` indicates that an CICS Service Flow Runtime module reported the error. For example, the CICS SFR interface program, DFHMADPL, or the Navigation Manager, DFHMAMGR.

Activity
The activity name that reported the error or where the failure occurred. The activity name is the program name being run as an activation. If the error was reported in the CICS SFR interface program, DFHMADPL, this field is blank. For information on activity names, see Developing with the BTS API.

Node Name
The node name that reported the error or where the failure occurred. The node name is the program name being run as an activation. Node names are server adapters.

Event
The last reattach event retrieved at the time of failure. For information regarding BTS events, see Developing with the BTS API.

Event type
The event type of the last reattach event retrieved at the time of failure. For information regarding BTS events and event types, see Developing with the BTS API. Valid values are as follows:

- Input
- Activity
- Composite
- Timer
- System
- None
- Unknown

A value of Unknown might indicate that the data in the event type field is not allowed.

Step
Indicates the internal program step at the time of failure.

Proctype
The process type of the request or process that reported the error or failure.

Process
The process name of the request processing instance that reported the error or failure.

Failed Processtype
The failed process type of a previously run instance of the request processing that resulted in error or failure. This field is populated when an error or failure occurs in a service flow process that was run as the result of an earlier processing failure.

Failed Process
The failed process name of a previously run instance of the request processing that resulted in error or failure. This field is populated when an error or failure occurs in a service flow process that was run as the result of an earlier processing failure.

ReplyToQ
The WebSphere MQ reply queue name where CICS Service Flow Runtime writes the error reply. This field is populated when an error or failure occurs in asynchronous request processing.

For information regarding WebSphere MQ programming, see the IBM MQ product documentation.

ReplyToQMgr
The WebSphere MQ reply queue manager name where CICS Service Flow Runtime writes the error reply. This field is populated when an error or failure occurs in asynchronous request processing.

For information regarding WebSphere MQ programming, see the IBM MQ product documentation.

MQ MsgId
The WebSphere MQ message identifier set by the service requester for use on all **MQPUT** commands in the queue server adapter and reply messages. This field is populated when an error or failure occurs in asynchronous request processing. For information regarding WebSphere MQ programming, see the IBM MQ product documentation.

MQ CorrelId
The WebSphere MQ correlation identifier set by the service requester. This field is populated when an error or failure occurs during asynchronous request processing.

For information regarding WebSphere MQ programming, see the IBM MQ product documentation.

Error detail
Brief description of the error that has occurred. This description is followed by specific error details that apply to the particular type of message that has been issued.

DFHMA000xx and DFHMA001xx installation error messages

An installation error message can occur during the restart of a CICS region where CICS SFR is installed.

DFHMA00001I CICS SFR flow installation has started.

Explanation:
The CICS Service Flow Runtime installation process has begun.

System action:
Processing continues.

User response:
None.

Destination:
Console.

DFHMA00002I CICS SFR flow installation completed. Successful: *ssss* Failed: *ffff*

Explanation:
The CICS Service Flow Runtime installation process has completed. The number of successfully installed flows is shown as *ssss* and the number of unsuccessfully installed flows as *ffff*. CICS SFR is now available for processing.

System action:
Processing continues.

User response

You can now invoke CICS SFR.

If *ffff* is greater than 0, look in the output for the CMAC transient data queue for messages that explain why the flow or flows have failed to install successfully.

Destination:
Console.

DFHMA00003I No CICS SFR installation was performed because an emergency restart was performed.

Explanation:
A CICS emergency start was performed. CICS SFR does no install processing during this type of CICS start up.

System action:
Processing continues.

User response

You can now invoke CICS SFR.

No install processing was done during start up. If any service flow properties files were added to the flow directory prior to the emergency startup, you must invoke a scan of the deployment directory to install them.

Destination:
Console.

DFHMA00004I No CICS SFR installation was performed because the NO_SFR_INST value was specified for the INITPARM system initialization parameter.

Explanation:
Because of the **INITPARM** value of NO_SFR_INST, no installation processing has taken place.

System action:
Processing continues. No CICS SFR installation processing will take place during this execution of CICS.

User response:
None.

Destination:
Console.

DFHMA00005I No CICS SFR installation was performed. No INITPARM value was specified for DFHMAINS.

Explanation:
DFHMAINS was invoked during PLTPI processing, but no **INITPARM** parameter was specified for it.

System action:
Processing continues. No CICS SFR installation processing will take place during this execution of CICS.

User response:
If you require CICS SFR processing, supply an **INITPARM** parameter for DFHMAINS and restart CICS.

Destination:
Console.

DFHMA00006E No CICS SFR installation was performed; INITPARM is not a valid directory name.

Explanation:
An **INITPARM** value was specified for DHFMAINS, but this value is a not a valid zFS directory name.

System action:
Any previously installed service flows are deleted and no flows are installed.

User response:
Correct the DFHMAINS **INITPARM** parameter and restart CICS to enable new service flows to be installed.

Destination:
Console.

DFHMA00007E CICS SFR installation failed. Unable to start transaction CMIT.

Explanation:

Because zFS processing is unavailable before control is given to CICS, the installation process is run by issuing a **START** of transaction CMIT during the PLTPI phase. This **START** command has failed.

System action:
Processing continues, but no CICS SFR installation processing takes place.

User response:
Determine why the CMIT transaction failed to start and correct the problem before restarting CICS or running the flow management transaction CMAN.

Destination:
Console.

DFHMA00100I Service flow properties directory: *zfs_directory*

Explanation:
The zFS directory to be scanned for service flow properties files, as specified in the DFHMAINS **INITPARM** parameter.

System action:
Processing continues.

User response:
None.

Destination:
CMAC transient data queue.

DFHMA00101I CICS SFR installing from file: *sfp_name*

Explanation:
A service flow properties file, *sfp_name*, has been found.

System action:
CICS SFR attempts to install the flow and resources as defined in the file *sfp_name*.

User response:
None.

Destination:
CMAC transient data queue.

DFHMA00110E Error changing to directory *zfs_directory*.

Explanation:
An attempt has been made to change to the directory *zfs_directory*. This attempt has failed.

System action:
CICS SFR installation fails. No new flows are installed. If this start is not a cold start, any previously installed flows will be available for invocation.

User response

See the following DFHMA00150E message to help determine the cause of the failure. Some reasons might be as follows:

- *zfs_directory* was not specified correctly.
- The incorrect case was used. zFS is case-sensitive.

Destination:
CMAC transient data queue.

DFHMA00111E Error opening directory *zfs_directory*

Explanation:
An attempt has been made to open the directory *zfs_directory*, but it has failed.

System action:
CICS SFR installation fails. No new flows are installed. If this start is not a cold start, any previously installed flows will be available for invocation.

User response:
See the following DFHMA00150E message to help determine the cause of the failure. The correct access rights might not have been configured to allow CICS to open this directory.

Destination:
CMAC transient data queue.

DFHMA00112E Error reading directory *zfs_directory*

Explanation:
An attempt has been made to read the directory *zfs_directory*, but it has failed.

System action:
CICS SFR installation fails. No new flows are installed. If this start s not a cold start, any previously installed flows will be available for invocation.

User response:
See the following DFHMA00150E message to help determine the cause of the failure. The correct access rights might not have been configured to allow CICS to read this directory.

Destination:
CMAC transient data queue.

DFHMA00113E Error closing directory *zfs_directory*.

Explanation:
An attempt has been made to close the directory *zfs_directory*, but it has failed.

System action:
CICS SFR installation processing has completed and all flows have been installed and are available.

User response:

See the following DFHMA00150E message to help determine the cause of the failure.

Destination:
CMAC transient data queue.

DFHMA00120E Error opening file *sfp_name*.

Explanation:
An attempt to open the service flow properties file *sfp_name* has failed.

System action:
Processing continues with the next file. The flow described by this file is not installed.

User response:
See the following DFHMA00150E message to help determine the cause of the failure.

Destination:
CMAC transient data queue.

DFHMA00121E Error reading file *sfp_name*.

Explanation:
An attempt to read the service flow properties file *sfp_name* has failed.

System action:
Processing continues with the next file. The flow described by this file is not installed.

User response:
See the following DFHMA00150E message to help determine the cause of the failure.

Destination:
CMAC transient data queue.

DFHMA00122E Error closing file *sfp_name*.

Explanation:
An attempt to close the service flow properties file *sfp_name* has failed.

System action:
Processing continues with the next file. The flow described by this file has been installed.

User response:
See the following DFHMA00150E message to help determine the cause of the failure.

Destination:
CMAC transient data queue.

DFHMA00123E Permission denied when trying to open *sfp_name*.

Explanation:
An attempt has been made to open file *sfp_name*, but the access rights have not been configured correctly to allow CICS to open the file.

System action:
Processing continues with the next file. The flow described by this file is not installed.

User response:
Correct the access rights and either use the flow management transaction, CMAN, to initiate a scan of the service flow directory or restart CICS.

Destination:
CMAC transient data queue.

DFHMA00124E File exceeds maximum of 32 600 bytes. File name: *sfp_name*

Explanation:
A service flow properties file has been found in the flow directory, but it exceeds the maximum size of 32 600 bytes.

System action:
Processing continues with the next file. The flow described by this file is not installed.

User response:
Regenerate the flow and deploy the file from service flow project tools again. If the problem persists, contact IBM.

Destination:
CMAC transient data queue.

DFHMA00125E File too small to be valid. File name: *sfp_name*.

Explanation:
A service flow properties file has been found in the flow directory, but it is too small to be a valid file.

System action:
Processing continues with the next file. The flow described in the file is not installed.

User response:
Regenerate and deploy the file from service flow project tools again. If the problem persists, contact IBM.

Destination:
CMAC transient data queue.

DFHMA00130E Unknown control block found in *sfp_name* at offset: *offset*

Explanation:
During the installation of the service flow in file *sfp_name*, invalid data was encountered at offset *offset*.

System action:
Processing continues with the next file. The service flow that is described in this file is not installed.

User response:
Redeploy the service flow from service flow project tools. If the problem persists, contact IBM.

Destination:
CMAC transient data queue.

DFHMA00140E Error reading DFHMAASF: *error*

Explanation:
During a cold start, the DFHMAASF file is read and each record is deleted. The read has failed for reason *error*.

System action:
Processing continues.

User response:
Determine the cause of the failure and correct it.

Destination:
CMAC transient data queue.

DFHMA00141E Unable to write record *reckey* to DFHMAASF.

Explanation:
After building the control blocks for service flow *reckey*, the attempt to write the record to file DFHMAASF has failed.

System action:
Processing continues with the next file. This service flow has not been installed.

User response:
None.

Destination:
CMAC transient data queue.

DFHMA00142E Unable to delete record *reckey* from DFHMAASF.

Explanation:
An attempt to delete a record, with the key=*reckey*, has failed.

System action:
Processing continues.

User response:
Using CICS logs and trace, determine the cause of the failure and take corrective action.

Destination:
CMAC transient data queue.

DFHMA00145E CICS SFR installation failed.

Explanation:
Installation processing has failed.

System action:
Processing ends.

User response:
Determine the cause of the error by searching for preceding messages or those written to the MSGUSR DD statement. Correct the problem before restarting CICS and using the flow management transaction, CMAN, to initiate scanning of the service flow deployment directory for new or updated service flow properties files.

Destination:
Console

DFHMA00150E A return code of *code* was issued with reason code *reason* for call *zfscall*

Explanation:
A zFS call has been made that has returned a nonzero return code. The previous DFHMA00xxxE error message describes the error. This message displays the return code and reason code associated with the error from the previous message. *zfscall* is the UNIX System Service issuing the nonzero return code.

System action:
Processing continues.

User response:
See the previous error message to ascertain what occurred. See z/OS UNIX System Services Messages and Codes for an explanation of the return code and reason code. Take corrective action based on this information. If the problem persists, contact IBM.

Destination:
CMAC transient data queue.

DFHMA002xx CMAN transaction messages

These messages are issued when you use the CMAN flow management transaction to install, enable, disable, and delete service flows.

DFHMA00200I Type action codes and press Enter, or press F2 to install flows

Explanation:
The CMAN transaction enables you to perform several actions on one or more service flows. The action codes are displayed by CMAN in the main menu.

System action:
None.

User response:
Perform one or more actions as appropriate.

Destination:
Console

DFHMA00201W Invalid key was pressed.

Explanation:

You entered an invalid key when performing an action on one or more service flows.

System action:
The function key is ignored.

User response:
Use the information in the menu display to enter the correct key. Press F1 if you require help.

Destination:
Console

DFHMA00202I CMAN session has completed.

Explanation:
The CMAN transaction has ended.

System action:
None.

User response:
None.

Destination:
Console, CMAC transient data queue

DFHMA00203I CICS is terminating.

Explanation:
The CMAN transaction has detected that CICS is terminating.

System action:
CMAN terminates.

User response:
None

Destination:
Console, CMAC transient data queue

DFHMA00204E A CMAN action must be supplied

Explanation:
The CMAN transaction was started from a console by entering CMAN without specifying an *action* or by another task using **EXEC CICS START TRANSID('CMAN')** without specifying the **FROM**(*action*) parameter.

System action:
The transaction terminates.

User response:
Either include an *action* or start CMAN from a terminal.

Destination:
CMAC transient data queue

DFHMA00205I No actions to take

Explanation:
You pressed the ENTER key without specifying an action.

System action:
Processing continues.

User response:
Specify an action and press ENTER.

Destination:
Console

DFHMA00206S CICS *command name* failed, RESP=*respcode*, RESP2=*resp2code* RCODE=*returncode*

Explanation:
The CMAN transaction receive an unexpected response when issuing an **EXEC CICS** command. The response is in *respcode*, *resp2code*, and *returncode*.

System action:
The CMAN transaction might continue or terminate, depending on what error occurred.

User response:
Look up the CICS RESP and RESP2 codes in CICS Application development reference to find out why CICS failed.

Destination:
Console, CMAC transient data queue

DFHMA00207S VSAM flow file DFHMAASF *action* failed RESP=*respcode* RESP2=*resp2code*

Explanation:
The CMAN transaction received an unexpected response when issuing an **EXEC CICS** command on VMSA file DFHMAASF. The response is in *respcode* and *resp2code*.

System action:
The transaction might terminate or continue, depending on the error that occurred.

User response:
For further details of the exception *respcode*, refer to CICS Application development reference or Reference: system programming.

Destination:
Console, CMAC transient data queue

DFHMA00208S CMAN has abended *abend_code* in program *program_name*

Explanation:
The CMAN transaction has abended. The abend code is *abend_code* and the failing program is *program_name*.

System action:
The transaction is terminated with a transaction dump with a dump code of *abend_code*.

User response:
Look up the abend code in CICS messages. For information on how to determine system problems, see Troubleshooting and support.

Destination:
Console, CMAC transient data queue

DFHMA00209E Unsupported type of CMAN task initiation

Explanation

The CMAN transaction has been initiated in a way that is not allowed. The only valid ways to initiate a CMAN transaction are:

- From a terminal
- From a console
- Issuing **`EXEC CICS START TRANSID('CMAN')`** from another task

System action:
The transaction terminates.

User response:
Use one of the methods above to initiate CMAN.

Console

DFHMA00210E Incorrect CMAN taction

Explanation

The CMAN transaction received an incorrect value for *action* when it was started from a terminal or console by entering CMAN *action*, or by another task using the **`EXEC CICS START TRANSID('CMAN') FROM(action)`**. The valid *action* values are:

- DELETE FLOW(*filter*)
- DISABLE FLOW(*filter*)
- ENABLE FLOW(*filter*)
- INSTALL

System action:
The transaction terminates.

User response:
Correct the *action* value.

Console, CMAC transient data queue

DFHMA00211E Adapter not found - action ignored.

Explanation:
During an action to change an adapter server, it was found that the adapter server is no longer present in the service flow repository file.

System action:
Action ignored.

User response:
None.

Console

DFHMA00212 Invalid action code.
W

Explanation:
You entered an invalid action code.

System action:
The action codes are not accepted because they are not all correct.

User response:
Correct the action code.

Console

DFHMA00213I Press Enter to confirm actions or F12 to cancel

Explanation:
You have entered action codes and then pressed F3.

System action:
None.

User response:
Either press F12 to cancel the actions and press F3 again, or press ENTER to confirm the actions and press F3 again.

Console

DFHMA00214I Install complete. Press Enter to update the display

Explanation:
Installation of service flows has completed successfully.

System action:
None.

User response:
Press Enter to update the display to show the newly installed service flows.

Console

DFHMA00215I All actions completed.

Explanation:
The CMAN transaction successfully completed all of the actions that you specified. The service flow repository file has been updated accordingly.

System action:
None.

User response:
None.

Destination:
Console

DFHMA00216I *number* service flows *actioned* using filter *filter*

Explanation:

An invocation of CMAN with action and filter parameters has completed. There were *number* number of service flows matched by the filter that were eligible for the specified action.

System action:
The requested actions have completed.

User response:
None.

Destination:
Console, CMAC transient data queue

DFHMA00217I Installation failed.

Explanation:
The installation of service flows has completed with errors.

System action:
None.

User response:
See the CICS message log for details of the errors.

Console, CMAC transient data queue

DFHMA00218I Installation complete

Explanation:
The installation of service flows has completed successfully.

System action:
None.

User response:
None.

Console, CMAC transient data queue

DFHMA010*xx* VSAM file error messages

The error messages contain the failed program name, transaction, file name, record key, and other relevant information. Use this information to assist in problem diagnosis.

As well as the standard set of fields that are available in each message, VSAM file error messages can also display the following information:

File name
The file name where the error occurred.

File function
The function that was attempted for the file that resulted in the error.

File length
The record length value of the VSAM KSDS file that resulted in the error.

File key data
The file key data value for the VSAM KSDS file that resulted in the error.

Key length
The file key length value for the VSAM KSDS file that resulted in the error

RESP and RESP 2 code
The CICS codes that were returned. See CICS Application development reference for CICS response codes and definitions.

DFHMA01001E FILE-READ-ERRMSG

Explanation

An **EXEC CICS READ** command to a VSAM file has failed in either a CICS Service Flow Runtime program, a generated flow navigator, or a server adapter.

System action:
Service flow processing ends.

User response

The error message contains the name of the program that issued the error. If the error occurred in the Link3270 business state file, DFHMAL2F, ensure that the Link3270 state cleanup tasks are not running. Use the CICS transaction ID from the error message to determine the program name of the generated Link3270 navigator.

Ensure that the Link3270 business state file is defined correctly and enabled to CICS.

Destination:
CMAC transient data queue.

DFHMA01002E FILE-RECTYPE-ERRMSG

Explanation:
A read to the service flow repository file, DFHMAASF, was successful. However, CICS SFR found an inconsistency in the definition of a service flow or the service flow has exceeded the maximum number of 200 subflows that is allowed.

System action:

Service flow processing ends.

User response:
The error message contains the name of the service flow that is causing the error. Discard and reinstall the service flow properties file for that service flow using the flow management transaction, CMAN. If the problem still occurs, regenerate the service flow from service flow project tools and redeploy it to CICS.

Destination:
CMAQ transient data queue.

DFHMA01003E FILE-STARTBR-ERRMSG

Explanation:
An **EXEC CICS STARTBR** command to a VSAM file has failed in a CICS Service Flow Runtime system module or in one of your generated FEPI navigator programs.

User response
The error message contains the program name that issued the error.

- If the error was issued by a FEPI flow navigator, the flow navigator failed to browse the alternate index SLU connection file, DFHMAC1F. The command uses the GENERIC and EQUAL options.
- If the error was issued by DFHMALFC, it failed to browse the Link3270 business state file, DFHMAL2F. The command uses the GETQ option. The Link3270 business state file system cleanup module, DFHMALFC, might have run to delete expired Link3270 facilities and associated state while performing request processing. This procedure is not recommended.

Check the file attributes and that the file is defined correctly and enabled to CICS.

Destination:
CMAC transient data queue.

DFHMA01004E FILE-READNXT-ERRMSG

Explanation:
An **EXEC CICS READNEXT** command to a VSAM file has failed in a CICS Service Flow Runtime program or in one of your generated FEPI navigator programs.

System action:
Service flow processing ends.

User response
The error message contains the program name that failed.

- If the error was reported by a FEPI flow navigator, a browse of the alternate index SLU Connection file, DFHMAC1F, has failed.
- If the error was reported by DFHMALFC, a browse with the GTEQ option specified to the Link3270 business state file DFHMAL2F has failed. The Link3270 business state file system cleanup module, DFHMALFC, might have run to delete expired Link3270 facilities and associated state while performing request processing. This procedure is not recommended.

Check the file attributes in the error message and that the file is defined correctly and enabled to CICS. Look up the CICS response codes for the READNEXT command and take appropriate actions.

Destination:
CMAC transient data queue.

DFHMA01005E FILE-REWRITE-ERRMSG

Explanation:
An **EXEC CICS REWRITE** command to a VSAM file has failed in a CICS Service Flow Runtime program or in a FEPI or Link3270 flow navigator.

System action:
Service flow processing ends.

User response
The error message contains the program name that issued the error.

- If the error was issued by a FEPI flow navigator, an error occurred when writing to the target interaction file, DFHMATIF or the SLU connection file, DFHMACOF.
- If the error was issued by DFHMALFS, an error occurred when writing to the Link3270 business state file, DFHMAL2F.

Check the file attributes and that the file is defined correctly and enabled to CICS. If the error occurred in DFHMAL2F, use the CICS transaction ID from the error message to determine the program name of the generated Link3270 flow navigator.

Destination:
CMAC transient data queue.

DFHMA01006E FILE-WRITE-ERRMSG

Explanation:
An **EXEC CICS WRITE** command to a VSAM file has failed in a CICS Service Flow Runtime program or in a FEPI or Link3270 flow navigator program.

System action:
Service flow processing ends.

User response
The error message contains the program name that failed.

- If a FEPI flow navigator issued the error, there was an error when writing to the target interaction file, DFHMATIF or the SLU connection file, DFHMACOF.
- If the DFHMALFS program issued the error, there was an error when writing to the Link3270 business state file DFHMAL2F.

Check the file attributes and that the specified file is defined correctly and enabled to CICS. If the error occurred in DFHMAL2F, use the CICS transaction ID from the error message to determine the program name of the Link3270 flow navigator that called the program reporting the problem.

Destination:
CMAC transient data queue.

DFHMA01007E FILE-ENDBR-ERRMSG

Explanation:
An **EXEC CICS ENDBR** command to a VSAM file has failed in a CICS Service Flow Runtime program or in a FEPI flow navigator.

System action:
Service flow processing ends.

User response
The error message contains the program name that reported the error.

- If the error was reported by a FEPI flow navigator, the flow navigator failed to end a browse of the alternate index SLU connection file, DFHMAC1F.
- If the error was reported by DFHMALFC, the program failed to end a browse of the Link3270 business state file, DFHMAL2f.

If the error occurred with DFHMAL2F, use the CICS transaction ID from the error message to determine the program name of the Link3270 flow navigator.

Destination:
CMAC transient data queue.

DFHMA01008E FILE-DELETE-ERRMSG

Explanation
An **EXEC CICS DELETE** command to a VSAM file has failed in a CICS Service Flow Runtime program or in a Link3270 flow navigator.

If reported by CICS SFR system modules, this condition occurred on the Link3270 business state file DFHMAL2F.

System action:
Service flow processing ends.

User response
Check the program name in the error message. If the error was reported by DFHMALFS, DFHMALFC, or DFHMALFD, there was a problem deleting from the Link3270 business state file, DFHMAL2F.

Check the attributes of DFHMAL2F and that it is defined correctly and enabled to CICS. Use the CICS transaction ID from the error message to determine the program name of the generated Link3270 flow navigator.

Destination:
CMAC transient data queue.

DFHMA013*xx* temporary storage queue (TSQ) error messages

Temporary storage queues (TSQs) are used for Link3270 server adapter processing only. The error messages contain the failed program name, transaction, TSQ name, CICS response codes, and other relevant information. Use this information to assist in problem diagnosis.

TSQs are used only when processing simple, nonpersistent service flows. TSQ error messages are reported in modules, DFHMALTS, DFHMALSC, and DFHMALFD.

If the error occurred in a Link3270 navigator, the error message contains the program name of the generated Link3270 navigator that called the system module, DFHMALTS.

DFHMA01331E TS-READ-ERRMSG

Explanation
A **READQ TS** command has failed in the CICS Service Flow Runtime or in the Link3270 State Cleanup program DFHMALSC.

See "Facility state cleanup processing" on page 71 for a description of how facility state information is managed.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA01332E TS-REWRITE-ERRMSG

Explanation:

A **WRITEQ TS** command with the REWRITE option has failed in either the CICS Service Flow Runtime or in the Link3270 State Cleanup program, DFHMALSC.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA01333E TS-WRITE-ERRMSG

Explanation

A **WRITEQ TS** command has failed in either the CICS Service Flow Runtime or in the Link3270 State Cleanup program, DFHMALSC.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA01334E TS-DELETE-ERRMSG

Explanation:
A **DELETEQ TS** command has failed in either the CICS Service Flow Runtime or in one of the Link3270 State Cleanup programs, DFHMALSC or DFHMALFD.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA01335E TS-INQSTART-ERRMSG

Explanation

An **INQUIRE TSQNAME START AT** command has failed in the Link3270 State Cleanup program, DFHMALSC. The command value is 'DFHMA' + LOW-VALUES.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA01336E TS-INQNEXT-ERRMSG

Explanation:
An **INQUIRE TSQNAME** command with the NEXT option has failed in the Link3270 State Cleanup program, DFHMALSC, with a condition other than END.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA01337E TS-INQEND-ERRMSG

Explanation:
An **INQUIRE TSQNAME** command with the END option has failed in the Link3270 State Cleanup program, DFHMALSC.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA020*xx* data-container error messages

The error messages contains the failed program name, transaction, container name, container owner, CICS response codes, and other relevant information. Use this information to assist in problem diagnosis.

DFHMA02001E GET-CONTAINER-ERRMSG

Explanation:
An **EXEC CICS GET CONTAINER** command has failed.

System action:
Service flow request processing ends.

User response

The error message contains the failed program name, transaction, container name, container owner, and CICS response codes. Look up the CICS RESP and RESP2 codes in CICS Application development reference to find out what error occurred.

If you cannot solve the problem, contact your IBM support center for assistance.

DFHMA02002E SET-CONTAINER-ERRMSG

Explanation:
An **EXEC CICS SET CONTAINER** command has failed.

System action:
Service flow request processing ends.

User response
The error message contains the failed program name, transaction, container name, container owner, and CICS response codes. Look up the CICS RESP and RESP2 codes in CICS Application development reference to find out what error occurred.

If you cannot solve the problem, contact your IBM support center for assistance.

DFHMA02003E PUT-CONTAINER-ERRMSG

Explanation:
An **EXEC CICS PUT CONTAINER** command has failed.

System action:
Service flow request processing ends.

User response
The error message contains the failed program name, transaction, container name, container owner, and CICS response codes. Look up the CICS RESP and RESP2 codes in CICS Application development reference to find out what error occurred.

If you cannot solve the problem, contact your IBM support center for assistance.

DFHMA02004E DELETE-CONTAINER-ERRMSG

Explanation:
An **EXEC CICS DELETE CONTAINER** command has failed.

System action:
Service flow request processing ends.

User response
The error message contains the failed program name, transaction, container name, container owner, and CICS response codes. Look up the CICS RESP and RESP2 codes in CICS Application development reference to find out what error occurred.

If you cannot solve the problem, contact your IBM support center for assistance.

DFHMA030*xx* Program link server adapter error messages

The error message contains the failed program name, transaction, target program name, CICS response codes, and other relevant information. Use this information to assist in problem diagnosis.

See CICS Application development reference for details of the **EXEC CICS LINK** command and CICS response codes.

DFHMA03001E DPL-ERRMSG

Explanation:
An **EXEC CICS LINK** command has failed. The most likely cause of this error is a PGMIDERR or SYSIDERR.

System action:
Service flow processing ends.

User response:
Use the information in the error message to check that the target program name and SYSID are correct.

Destination:
CMAC transient data queue.

DFHMA03003E DPL-LINK-ERRMSG

Explanation:
A value other than COMMAREA or CHANNEL has been requested.

System action:
Service flow request processing ends.

User response:
Use the information in the error message to check the activity name and the field that is in error. The field might be either DPC-COMMAREA or DPC-CHANNEL. Check the program link in the service flow model to ensure it specifies the correct value. Regenerate and redeploy the service flow.

Destination:
CMAC transient data queue.

DFHMA03004E DPL-CONTAINER-IND-ERRMSG

Explanation:
An application output container is not identified as either optional or required.

System action:
Service flow request processing ends.

User response:
Use the information in the error message to find the name of the container that is reporting the error. The field can either be 0 (optional) or R (required). Ensure that all of the containers in the service flow model are defined as required or optional, and then regenerate and redeploy the service flow.

Destination:
CMAC transient data queue.

DFHMA03005E DPL-REQUEST-NO-ERRMSG

Explanation:

The number of application input containers is invalid. The valid range is 0 to 999.

System action:
Service flow request processing ends.

User response:
The error message displays the number of containers on the program link request. If it is greater than 999, remodel the service flow to use 999 containers or fewer. Regenerate and deploy the service flow.

Destination:
CMAC transient data queue.

DFHMA03006E DPL-RESPONSE-NO-ERRMSG

Explanation:
The number of application output containers is invalid. The valid range is 0 to 999.

System action:
Service flow request processing ends.

User response:
The error message displays the number of containers on the response to the program link. If it is greater than 999, remodel the service flow to use 999 containers or fewer. Regenerate and deploy the service flow.

Destination:
CMAC transient data queue.

DFHMA040*xx* FEPI server adapter messages

FEPI server adapter messages are written to the CMAC transient data queue. Error messages contain the failed program name, transaction, target program name, target APPLID, conversation ID, CICS response codes, External Security Manager (ESM) return codes, and other relevant information. Use this information to assist in problem diagnosis.

See CVDA and RESP2 values for FEPI commands for CICS response codes and definitions. You can find additional diagnostic information in the FEPI message log.

DFHMA04001E PASSTICKET-ERRMSG

Explanation:
An **EXEC CICS FEPI REQUEST PASSTICKET** command has failed. The most likely cause of failure is an external security manager error.

System action:
Service flow processing ends.

User response:
For response and reason codes returned by RACF, see z/OS Security Server RACF Messages and Codes.

Destination:
CMAC transient data queue.

DFHMA04002E ALLOCATE-ERRMSG

Explanation:
An **EXEC CICS FEPI ALLOCATE POOL** or **ALLOCATE PASSCONVID** command has failed. The most likely cause of this error if you are attempting to acquire ownership of an existing unowned conversation, **ALLOCATE PASSCONVID**, is that the conversation ID is not known or the conversation has been lost. When it attempts to establish a new conversation with a target application, **ALLOCATE POOL**, the most likely reason why this message occurs is the POOL or TARGET or both are unavailable, not known, or no session is available and in service.

System action:
Service flow processing ends.

User response:
Check the error message details to determine the POOL and TARGET. Check if the POOL and TARGET are defined, installed, and in service. Also check if in-service sessions are available for the defined TARGET.

Destination:
CMAC transient data queue.

DFHMA04003E SEND-ERRMSG

Explanation:
An **EXEC CICS FEPI SEND DATASTREAM** command with the INVITE option has failed. The most likely cause of this error is a lost session or other session state violation.

System action:
Service flow processing ends.

User response:
Check the details in the error message to find out what caused the problem.

Destination:
CMAC transient data queue.

DFHMA04004E RECEIVE-ERRMSG

Explanation:
An **EXEC CICS FEPI RECEIVE DATASTREAM** command has failed. The command options used are `UNTILCDEB` and `MAXFLENGTH(25000)`. The most likely cause of this error is a session lost or other session state violation.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem. Check that the maximum length of the data returned is not greater than 25 000 bytes.

Destination:
CMAC transient data queue.

DFHMA04005E EXTRACT-ERRMSG

Explanation:
An **EXEC CICS FEPI EXTRACT CONV** command has failed. This command is issued after a successful **ALLOCATE POOL** command and after an unsuccessful **RECEIVE DATASTREAM**, **SEND DATASTREAM**, or **CONVERSE DATASTREAM** command to retrieve the POOL, TARGET, NODE, and SENSEDATA. The most likely cause of this error is that a session has been lost.

System action:
Service flow processing ends.

User response:
Use the error information in the message to determine the problem. Check that the maximum length of the data returned is not greater than 25 000 bytes.

Destination:
CMAC transient data queue.

DFHMA04006E FREE-ERRMSG

Explanation

An **EXEC CICS FEPI FREE** command has failed. The option used on this command depends on the value of the log off option, also known as the exit type, in your generated FEPI server adapter. This field can have the following values:

- R = RELEASE
- F = FORCE
- P = PASS
- H = HOLD

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine which FEPI server adapter is failing. Use the CMAN transaction to look up the value of the exit type for that FEPI server adapter.

Destination:
CMAC transient data queue.

DFHMA04007E SET-ERRMSG

Explanation:
An **EXEC CICS FEPI SET DATASTREAM** command has failed.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA04008E ISSUE-ERRMSG

Explanation:
An **EXEC CICS FEPI FREE** command has failed. An attempt was made to send a NORMALRESP as specified by the VALUE option to the target.

System action:
Service flow processing ends.

User response:
Use the information in the error message to determine the problem.

Destination:
CMAC transient data queue.

DFHMA04009E INQUIRE-ERRMSG

Explanation:
An **EXEC CICS FEPI INQUIRE POOL** command has failed. The most likely cause of this error is that the POOL name is not known.

System action:
Service flow processing ends.

User response:
Check that the POOL specified for the program reporting the error is correct. You can find this information in the error message. Make sure the POOL is defined and installed.

Destination:
CMAC transient data queue.

DFHMA04010E MAP3270-ERRMSG

Explanation:
A severe error has occurred in assembler module DFHMAFS0, entry point BLD3270I, when attempting to build an inbound data stream.

System action:
Service flow processing ends.

User response:
Take a CICS auxiliary trace using the AP and SZ components with a trace level of 1. The AP trace shows the service flow level traces and the SZ trace shows the FEPI traces. Contact IBM support, providing this tracing, to assist in problem diagnosis.

Destination:
CMAC transient data queue.

DFHMA04011E PROPERTY-ERRMSG

Explanation
The inquired POOL name has definitions that are not allowed or it is not in the correct state for use in the FEPI flow navigator. This problem can be caused by any of the following conditions:

- A FORMAT definition is not equal to DATASTREAM.
- A MAXFLENGTH definition is greater than 25 000 bytes.
- The install state, INSTLSTATUS, is not equal to INSTALLED.
- The service state, SERVSTATUS, is not equal to INSERVICE.

System action:
Service flow processing ends.

User response
Check the POOL resource definitions and state. You can check the POOL definitions by inquiring on the associated installed PROPERTYSET. Use the **CEMT INQ FEPOOL** command to view the state of the identified pool.

See the **INQUIRE POOL** command in FEPI system programming reference for more information.

Destination:
CMAC transient data queue.

DFHMA04012E RECEIVETRUNC-ERRMSG

Explanation:
An **EXEC CICS FEPI RECEIVE DATASTREAM** command has failed to receive the entire 3270 data stream for a screen. The command options used as UNTILCDEB and MAXFLENGTH.

System action:
The FEPI server adapter fails and service flow processing ends.

User response:
Increase the MAXFLENGTH for the FEPI pool. The maximum length is 25 000. Alternatively, use another pool for the terminal type with a larger MAXFLENGTH value.

Destination:
CMAC transient data queue.

DFHMA050*xx* Queue server adapter error messages

The error message contains the failed program name, transaction, queue name, queue manager name, completion codes, reason codes, and other relevant information. Use this information to assist in problem diagnosis.

See Troubleshooting and support in IBM MQ product documentation for completion and reason code values and definitions.

DFHMA05001E MQOPEN-ERRMSG

Explanation:
An **MQOPEN** call failed in the Queue server adapter.

System action:
Service flow processing ends.

User response
Check the error message to determine the problem. Ensure that the queue manager connection exists and that the queue manager is not quiescing.

Check the queue definition to ensure that the queue is defined as SHARE and DEFSOPT(SHARED) if more than one process is accessing the queue at the same time.

Check the queue that the program is attempting to open exists. The queue and queue manager names are specified in the service flow properties file.

Destination:
CMAC transient data queue.

DFHMA05002E MQPUT1-ERRMSG

Explanation:
An **MQPUT1** call has failed. This error might be reported on an **MQPUT** call to the reply queue specified in the DFHMAH header by DFHMADPL and DFHMAMGR, when attempting to issue reply messages to the service requester.

System action:
Service flow processing ends.

User response:
First check the queue attributes, such as PUT(DISABLED), CURDEPTH, and MAXMSGL. Also check the error message to determine what program reported the error. If the error was reported by the Queue server adapter, ensure that the local queue that the server adapter is attempting to send a message to exists. The queue name is specified in the service flow properties file and can be viewed in service flow project tools.

Destination:
CMAC transient data queue.

DFHMA05004E MQGET-ERRMSG

Explanation
An **MQGET** call has failed or returned a warning in the Queue server adapter.

The Queue server adapter uses the following options on the **MQGET** command when attempting to read messages off a queue:

- MQGMO-WAIT
- MQGMO-SYNCPOINT
- MQGMO-ACCEPT-TRUNCATED-MSG

When the Queue server adapter is processing synchronously, the **MQGET** command program uses MQGMO-NO-SYNCPOINT.

In addition, MatchOptions is set to MQMO-MATCH-CORREL-ID. The correlation ID value is the message ID field, if there is a value in the DFHMAH header of the request message. Otherwise, this value is the MsgId of the queue manager that is specified on the **MQPUT** command that is issued by the Queue server adapter.

System action:
Service flow processing ends.

User response
Check the error message to determine the problem. Ensure the queue manager connection exists and that the queue manager is not quiescing. Also check the queue attributes and **MQGET** wait interval. The wait interval is specified in the service flow properties file and can be viewed in service flow project tools.

You might also check the queue to see that it is not GET(DISABLED). Check that the reply queue and reply queue manager are correctly specified in the service flow. Check that your back-end application is correctly setting the CorrelId.

Destination:
CMAC transient data queue.

DFHMA05005E MQCLOSE-ERRMSG

Explanation:
An **MQCLOSE** call has failed.

System action:
Service flow processing ends.

User response:
Check the error message details to determine the problem. Ensure that the queue manager and the queue reported in the error are still available and that CICS is still connected to the queue manager.

Destination:
CMAC transient data queue.

DFHMA060*xx* BTS error messages

The error message contains the failed process ID, process type, program name, transaction, CICS BTS response codes, and other relevant information. Use this information to assist in problem diagnosis.

See Developing with the BTS API and CICS Application development reference for response codes and definitions.

DFHMA06001E DEFINE-PROCESS-ERRMSG

Explanation:
An **EXEC CICS DEFINE PROCESS** command has failed in DFHMADPL, when attempting to define a BTS process with the Navigation Manager as its root activity.

System action:
Service flow processing ends.

User response
Check that the Navigation Manager program DFHMAMGR and transaction CMAM are defined and enabled to CICS. In addition check the following:

- Check that the PROCESSTYPE is defined and enabled to CICS and that the PROCESS value is valid. The PROCESSTYPE has the same value as the request name of the service flow.
- Check that the process name value is not already in use. If it is, your method of allocating a unique process name might be in error, or an earlier process using the same name might have failed and be in an incomplete state. Use the CICS-supplied transaction CBAM to see which processes are defined for the PROCESSTYPE.

Destination:
CMAC transient data queue.

DFHMA06002E DEFINE-ACTIVITY-ERRMSG

Explanation:
An **EXEC CICS DEFINE ACTIVITY** has failed in DFHMAMGR or in a flow navigator program when attempting to define an activity. The most likely cause of this error is the target program name or transaction are not defined to CICS.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction ID used in the **DEFINE ACTIVITY** command. Check that this program and transaction

are defined and enabled to CICS. In addition, ensure that all your generated programs and transactions that are required to run the service flow are defined and enabled in the CICS region.

Destination:
CMAC transient data queue.

DFHMA06003E RUN-PROCESS-ERRMSG

Explanation:
An **EXEC CICS RUN ACQPROCESS** command has failed in the CICS SFR interface program DFHMADPL, when attempting to run a BTS process with the Navigation Manager as its root activity.

System action:
Service flow processing ends.

User response:
Use the information in the message to check that the request ran in synchronous mode, and that the transaction and program are available in the CICS region. The problem might be in the service flow repository file.

Destination:
CMAC transient data queue.

DFHMA06004E RUN-ACTIVITY-ERRMSG

Explanation:
An **EXEC CICS RUN ACTIVITY** command has failed in DFHMAMGR or a flow navigator when attempting to run an activity.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction. Check that the program and transaction definitions that implement the activity are correct. In addition, check that the user associated with the issuing task is authorized to run the activity and that the activity named on the command has not abended.

Destination:
CMAC transient data queue.

DFHMA06005E CHECK-PROCESS-ERRMSG

Explanation

This error is reported by the CICS SFR interface program DFHMADPL. One of the following conditions exists:

- The **EXEC CICS CHECK ACQPROCESS** command has failed.
- The acquired process was forced to complete, potentially using a **CANCEL ACQPROCESS** command that was issued outside CICS Service Flow Runtime.
- The Navigation manager, DFHMAMGR, abended.

User response:
Check the process status using the CBAM transaction . If BTS auditing was turned on for this PROCESSTYPE, dump the audit log to help determine the reason for the failure. In addition, check the CICS log for diagnostics assistance. If the problem is an abend, check the abend code in CICS messages for the reason and appropriate actions.

Destination:
CMAC transient data queue.

DFHMA06006E CHECK-ACTIVITY-ERRMSG

Explanation

This error is reported by Navigation manager (DFHMAMGR) and your generated adapter navigator programs. One of the following conditions exists:

- The **EXEC CICS CHECK ACTIVITY** command has failed.
- The acquired activity was forced to complete, potentially using a **CANCEL ACTIVITY** issued outside CICS Service Flow Runtime.
- The activity specified on the command abended.

System action:
Service flow processing ends.

User response:
Check the process status using the CBAM transaction. If BTS auditing was turned on for this PROCESSTYPE, dump the audit log to help determine the reason for failure. In addition, check the CICS log for diagnostics assistance. If the problem is an abend, check the abend code in CICS messages for the reason and appropriate actions.

Destination:
CMAC transient data queue.

DFHMA06007E ACQUIRE-PROCESS-ERRMSG

Explanation:
An **EXEC CICS ACQUIRE PROCESS** command has failed in DFHMADPL while attempting to acquire a previously failed CICS Service Flow Runtime process.

System action:
Service flow processing ends.

User response:
Check the error message to determine that the failed process type and process name values found in the DFHMAH header structure of the request message are specified and valid, FAILED-PROCTYPE and FAILED-PROCNAME respectively. These values were returned from a previously failed CICS Service Flow Runtime process. Check that the PROCESSTYPE resource is defined and enabled to CICS. Use the

CBAM transaction to determine that the process exists and its state.

Destination:
CMAC transient data queue.

DFHMA06011E CANCEL-PROCESS-ERRMSG

Explanation:
An **EXEC CICS CANCEL ACQPROCESS** command has failed in the CICS SFR interface program DFHMADPL, while attempting to cancel an acquired process that has failed. The most likely cause of this error is that the process mode is not allowed, or one or more of the activities in the process are inaccessible or are in CANCELLING mode.

System action:
Service flow processing ends.

User response:
Use the CBAM transaction to determine the process state.

Destination:
CMAC transient data queue.

DFHMA06017E RETRIEVE-EVENT-ERRMSG

Explanation:
An **EXEC CICS RETRIEVE REATTACH EVENT** command has failed in the Navigation manager (DFHMAMGR), a flow navigator, or a server adapter.

System action:
Service flow processing ends.

User response:
Contact your IBM support center for assistance.

Destination:
CMAC transient data queue.

DFHMA06018E DEFINE-EVENT-ERRMSG

Explanation:
An **EXEC CICS DEFINE INPUT EVENT** command has failed in the Navigation manager program, DFHMAMGR. While processing in asynchronous mode, an error occurred in either the Navigation manager program, a flow navigator, or a server adapter while processing the service flow. CICS Service Flow Runtime attempts to define an input event in the Navigation Manager after recognizing this error and the DEFINE INPUT EVENT command failed.

System action:
Service flow processing ends.

User response:
Contact your IBM support center for assistance.

Destination:
CMAC transient data queue.

DFHMA06021E INQUIRE-PROCESSTYPE-ERRMSG

Explanation

This error is reported by the CICS SFR interface program, DFHMADPL. One of the following conditions exist:

- The specified process type name is not correct.
- The specified PROCESSTYPE resource has not been installed in the CICS system.
- If no process type has been specified in the container, a PROCESSTYPE resource with the named service flow has not been installed.

System action:
Service flow processing ends.

User response:
Check that the specified PROCESSTYPE name is correct and installed in the CICS system.

Destination:
CMAC transient data queue.

DFHMA06022E ENABLED-PROCESSTYPE-ERRMSG

Explanation:
This error is reported by the CICS SFR interface program, DFHMADPL. The specified PROCESSTYPE resource is not ENABLED in the CICS system.

System action:
Service flow processing ends.

User response:
Check that the specified PROCESSTYPE resource is installed and enabled in the CICS system.

Destination:
CMAC transient data queue.

DFHMA070*xx* Link3270 server adapter error messages

The error message contains the failed program name, transaction response codes, and other relevant information. Use this information to assist in problem diagnosis.

See CICS Application development reference for CICS response codes and definitions, and Developing for external interfaces for Link3270 bridge header structure (BRIH), inbound and outbound vector message structure (BRIV) definitions, return, and completion codes.

DFHMA07001E DFHL3270-ERRMSG

Explanation:
An **EXEC CICS LINK** command for Link3270 bridge program, DFHL3270, has failed.

System action:
Service flow processing ends.

User response:
Check the CICS RESP and CICS RESP2 fields in the error message. Look up the values of the response codes in CICS Application development reference to determine what the error is.

Destination:
CMAC transient data queue.

DFHMA07002E DFHL3270-BRIH-ERRMSG

Explanation:
An error was reported by the Link3270 bridge program, DFHL3270, in the Link3270 bridge header structure, BRIH, fields BRIH-RETURNCODE, BRIH-COMPCODE, and BRIH-REASON. This error message might result from an error in postinstallation steps, with a Link3270 bridge error of transaction not found. The value of the BRIH-RETURNCODE field is 85.

System action:
Service flow processing ends.

User response

Check the following fields in the error message for the program reporting the error:

- Return code
- Comp code
- Reason code

See CICS Application development reference for specific return code values and actions relating to BRIH errors.

If this error indicates that a transaction is not found, a transaction definition required for AOR routing might be defined incorrectly or missing. See the Transactionid field to determine the invalid transaction ID. If the transaction ID value is CMAI or the transaction ID is equal to the CICS system name (SYSID) where your target CICS application transactions are running, you might have an incorrect AOR routing configuration. See “Configuring the runtime environment to use transaction routing” on page 70 for additional information.

Destination:
CMAC transient data queue.

DFHMA07010E NO-MAPNAME-ERRMSG

Explanation:
A SEND MAP outbound vector, 1804, was received in a Link3270 server adapter without a BMS map name in the BRIV outbound vector header field, BRIV-SM-MAP.

System action:
Service flow processing ends.

User response:
Check the Mapset and Map fields in the error message. Also check the BRIV outbound vector application data structure data in the error message and the target CICS application program to determine the problem.

Destination:
CMAC transient data queue.

DFHMA07011 PROTECTED-UPDATE-WARNING W

Explanation:
A Link3270 server adapter tried to update a protected map field with data in the BRIV inbound message vector Application Data Structure (ADS).

System action:
Service flow request processing ends.

User response:
Check the Field name field in the error message. Modify your service flow that contains the program reporting the error and redeploy it to CICS.

Destination:
CMAC transient data queue.

DFHMA07012E UNEXPECTED-VECTOR-ERRMSG

Explanation:
During transaction routing processing, a SEND outbound vector, 0404, was not received in a Link3270 server adapter from the DFHMALIN program.

System action:
Service flow processing ends.

User response

The error message contains the program name. Check the field SYSID for the program reporting the error. Check the CMAI transaction definition in the specified system name of the CICS server region to ensure the following items:

- The transaction is defined.
- The definition indicates the correct program, DFHMALIN.

Destination:
CMAC transient data queue.

DFHMA07013E NO-VECTOR-ERRMSG

Explanation:

No BRIV outbound vector application data structure data was received in a Link3270 server adapter when one was expected.

System action:
Service flow processing ends.

User response:
Check the error message for the Link3270 bridge header structure, BRIH, and BRIV outbound vector header control fields to determine what was sent. Check and possibly modify the program and name of the service flow and redeploy it to CICS. Also, check the target CICS application program to determine possible processing problems or unsupported functions.

Destination:
CMAC transient data queue.

DFHMA07014E INQUIRE-TRANSID-ERRMSG

Explanation:
An **EXEC CICS INQUIRE TRANSACTION** command has failed. The most likely cause is that a transaction is not defined in the CICS region.

User response
Check the following fields in the error message for the program reporting the error:

- Transid
- CICS Resp
- CICS Resp2

Check that the transaction identified in the error message is defined in the CICS region.

Destination:
CMAC transient data queue.

DFHMA07015 INVALID-ATTRIBUTE-WARNING W

Explanation:
An invalid attribute was found in the BRIV outbound vector Application Data Structure data. The attribute value is set to LOW-VALUES. The most likely cause is a map definition error or target CICS application program error.

System action:
Service flow processing continues.

User response
Check the following fields in the error message:

- Field name
- Invalid attribute

Check that the map and map field definitions are valid and supported. Check the target CICS application program for processing errors or unsupported functions.

Destination:
CMAC transient data queue.

DFHMA07016E MAPSET-LOAD-ERRMSG

Explanation:
An **EXEC CICS LOAD PROGRAM** command for the specified map set load module has failed. The most likely cause is that the map set load module is not defined in the CSD or the RPL concatenation.

System action:
Service flow processing ends.

User response:
Check the Mapset field in the error message for the program reporting the error. Update the CSD or RPL concatenation as necessary to define the map set load module in the CICS region.

Destination:
CMAC transient data queue.

DFHMA07017E MAP-NOT-FOUND-ERRMSG

Explanation:
The specified map was not found in the map set load module. The most likely cause is an incorrect map set load module.

System action:
Service flow processing ends.

User response:
Check the Mapset and Map fields in the error message for the program reporting the error. Check the RPL concatenation to determine if there is a program or map set load module of the same name, as specified in the Mapset field, higher in the concatenation. Also check that the correct map set load module is defined to the CICS region.

Destination:
CMAC transient data queue.

DFHMA07018E ADS-DESCRIPTOR-ERRMSG

Explanation:
An ADS descriptor is not present in the mapset load module. ADS descriptors are required in mapset load modules that are used with Link3270 bridge.

System action:
Service flow processing ends.

User response:
Reassemble the mapset to include an ADS descriptor. If you do not have the source, use the DFHBMSUP utility to recreate source statements from your mapset load module. See CICS utility programs for information about this utility.

DFHMA07901E UNSUPPORTED-VECTOR-ERRMSG

Explanation:
An unsupported BRIV outbound vector was received in the Link3270 vector processing module, DFHMAVCP. This error message might occur when the service flow is interacting with an application data structure (ADS) rather than a screen buffer, but the outbound vectors are SEND or CONVERSE vectors.

System action:
Service flow processing ends.

User response:
Modify the service flow to interact with the target application using a screen buffer instead of an ADS.

Destination:
CMAC transient data queue.

DFHMA080*xx* error messages

The error message contains the failed program name, transaction, CICS response codes, and other relevant information. Use this information to assist in problem diagnosis.

DFHMA08001E EIBCALEN-ERRMSG

Explanation:
An attempt was made to run DFHMADPL from a service requester with an EIBCALEN equal to zero.

System action:
Service flow processing ends.

User response:
Fix the application that is acting as the service requester to pass the correct structure to the CICS SFR interface program, DFHMADPL. DFHMADPL expects the DFHMAH header structure, followed by the request data when using a COMMAREA, and the request name when using a channel and containers.

Destination:
CMAC transient data queue.

DFHMA08002E COMMAREA-ERRMSG

Explanation:
The length of the application data, as specified in field DFHMAH-DATALENGTH of the message header structure DFHMAH, was greater than the length of the mapped input request area in the flow navigator program. Alternatively, the value of DFHMAH-DATALENGTH and DFHMAH-STRUCLENGTH exceeded EIBCALEN in the CICS SFR interface program, DFHMADPL.

System action:
Service flow processing ends.

User response:
Fix the service requester to pass the correct length of application data or correct the service flow and redeploy it to CICS.

Destination:
CMAC transient data queue.

DFHMA08006E CICS-LEVEL-ERRMSG

Explanation:
The CICS region is not at CICS Transaction Server Version 3.2, or higher.

System action:
Service flow processing ends.

User response:
Upgrade to the correct version and release of CICS.

Destination:
CMAC transient data queue.

DFHMA08008E FEJBDTRN/E-ERRMSG

Explanation:
An attempt to perform a BIDI transformation failed.

System action:
Service flow processing ends.

User response

Check the following fields in the error message:

- Response code
- Reason code

A response code of 5 indicates an incorrect BIDI transformation attribute string. The reason code indicates whether the attribute problem is one of the following:

- A duplicate reference (value 1)
- No Text Type attribute provided (value 2)
- No Orientation attribute (value 3)

The BIDI transformation attributes need updating using either the model in service flow project tools or, if dynamic messages are used, check the attributes in the request message.

The other response codes indicate that the system functions used to support the BIDI transformation have failed. The reason code provides the error code returned by the failing system function. These codes are for diagnostic purposes for the IBM Service team.

Destination:
CMAC transient data queue.

DFHMA08010E DATA-LENGTH-ERRMSG

Explanation:

The length of data in a channel container passed to DFHMADPL was outside the allowed range.

System action:
Service flow processing ends.

User response:
Correct the service requester to pass the correct channel container contents to DFHMADPL.

Destination:
CMAC transient data queue.

DFHMA08011E INPUT-COMBINATION-ERRMSG

Explanation:
An incorrect combination of channel containers was passed to DFHMADPL.

System action:
Service flow processing ends.

User response:
Correct the service requester to pass the correct container combinations on the channel to DFHMADPL.

Destination:
CMAC transient data queue

DFHMA081*xx* API error messages

These error messages result from failed CICS API commands. The error message contains the failed program name, transaction, CICS response codes, and other relevant information. Use this information to assist in problem diagnosis.

See CICS Application development reference for the particular CICS API command and CICS response codes. Each error message corresponds to a specific CICS API command that resulted in failure.

DFHMA08102E CICS-RETRIEVE-ERRMSG

Explanation
An **EXEC CICS RETRIEVE** command failed in the CICS Service Flow Runtime.

This error can occur in the following programs:

DFHMALSC
DFHMALSC is a program that cleans up Link3270 state information for simple, nonpersistent services flows, where state information is stored, retrieved, and deleted in temporary storage queues.

DFHMALFC
DFHMALFC is a program that cleans up the state information for simple, persistent service flows and complex service flows, where state information is stored, retrieved, and deleted in the Link3270 VSAM State file, DFHMAL2F.

DFHMALFD
DFHMALFD is a program that deallocates Link3270 facilities. It is invoked by a **CICS START** command from the following CICS SFR programs:

- DFHMALSC
- DFHMALFC.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate actions.

Destination:
CMAC transient data queue.

DFHMA08103E CICS-START-ERRMSG

Explanation
An **EXEC CICS START** command failed in the CICS Service Flow Runtime.

This error can occur in the following programs:

DFHMALSC
DFHMALSC is a program that cleans up the state information for simple, nonpersistent service flows, where state information is stored, retrieved, and deleted in temporary storage queues.

DFHMALFC
DFHMALFC is a program that cleans up the state information for simple, persistent service flows and complex service flows, where state information is stored, retrieved, and deleted in the Link3270 VSAM state file, DFHMAL2F.

Both DFHMALSC and DFHMALFC programs schedule and reschedule task STARTs using their transaction IDs, with an INTERVAL as defined in the terminal data when initially started by terminal input or as defined in the **INITPARM** system initialization parameter of the region. DFHMALSC transaction ID is CMAK. DFHMALFC transaction ID is CMAF. Each module can also schedule STARTs of transaction CMAD, when Link3270 facilities might have to be deallocated.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed

and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate actions.

Destination:
CMAC transient data queue.

DFHMA08104E CICS-ASSIGN-ERRMSG

Explanation:
An **EXEC CICS ASSIGN** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate actions.

Destination:
CMAC transient data queue.

DFHMA08106E CICS-ENQUEUE-ERRMSG

Explanation

An **EXEC CICS ENQ** command failed in CICS Service Flow Runtime. This error can occur in the following programs:

DFHMALSC
DFHMALSC is a program that cleans up state information for simple, nonpersistent service flows, where state information is stored, retrieved, and deleted in temporary storage queues.

DFHMALTS
DFHMALTS is a program that stores, retrieves, and deletes Link3270 state information for simple, nonpersistent service flows, where state information is stored, retrieved and deleted in Link3270 temporary storage queues.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate actions.

Destination:
CMAC transient data queue.

DFHMA08107E CICS-DEQUEUE-ERRMSG

Explanation

An **EXEC CICS DEQ** command failed. This error can occur in the following CICS SFR programs:

DFHMALTS
DFHMALTS is a program that stores, retrieves, and deletes Link3270 state information simple, nonpersistent service flows, where state information is stored, retrieved, and deleted in Link3270 temporary storage queues.

DFHMALSC
DFHMALSC is a program that cleans up Link3270 state information for complex, nonpersistent service flows, where state information is stored, retrieved, and deleted in temporary storage queues.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate actions.

Destination:
CMAC transient data queue.

DFHMA08108E CICS-INQUIRE-ERRMSG

Explanation:
An **EXEC CICS INQUIRE** command failed in the CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA08109E CICS-GETMAIN-ERRMSG

Explanation:
An **EXEC CICS GETMAIN** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue.

DFHMA08110E CICS-FREEMAIN-ERRMSG

Explanation:
An **EXEC CICS FREEMAIN** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue.

DFHMA08111E CICS-SOAPFAULT-ERRMSG

Explanation:
An **EXEC CICS SOAPFAULT CREATE** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the program name and transaction in which the CICS API command failed and the associated CICS response codes. Look up the response codes in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue.

DFHMA08112E CICS-INVOKEWS-ERRMSG

Explanation:
An **EXEC CICS INVOKE WEBSERVICE** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the associated CICS response codes for the failure with the CICS API command. Look up the response codes for the **EXEC CICS INVOKE WEBSERVICE** command in CICS Application development reference and take appropriate action.

Destination:
CMAQ transient data queue.

DFHMA08113E CICS-SEND-MAP-ERRMSG

Explanation:
An **EXEC CICS SEND MAP** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the CICS response code for the failure with the CICS API command. Look up the response codes for the **EXEC CICS SEND MAP** command in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA08114E CICS-RECEIVE-MAP-ERRMSG

Explanation:
An **EXEC CICS RECEIVE MAP** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the CICS response code for the failure with the CICS API command. Look up the response codes for the **EXEC CICS RECEIVE MAP** command in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA08115E CICS-ROUTE-ERRMSG

Explanation:
An **EXEC CICS ROUTE** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the CICS response code for the failure with the CICS API command. Look up the response codes for the **EXEC CICS ROUTE** command in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA08116E CICS-SEND-TEXT-ERRMSG

Explanation:
An **EXEC CICS SEND TEXT** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the CICS response code for the failure with the CICS API command. Look up the response codes for the **EXEC CICS SEND TEXT** command in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA08117E CICS-SEND-PAGE-ERRMSG

Explanation:
An **EXEC CICS SEND PAGE** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the CICS response code for the failure with the CICS API command. Look up the response codes for the **EXEC CICS SEND PAGE** command in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA08118E CICS-SEND-CTRL-ERRMSG

Explanation:
An **EXEC CICS SEND CONTROL** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the CICS response code for the failure with the CICS API command. Look up the response codes for the **EXEC CICS SEND CONTROL** command in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA08119E CICS-PURGE-MSG-ERRMSG

Explanation:
An **EXEC CICS PURGE** command failed in CICS Service Flow Runtime.

System action:
Service flow processing ends.

User response:
The error message contains the CICS response code for the failure with the CICS API command. Look up the response codes for the **EXEC CICS PURGE** command in CICS Application development reference and take appropriate action.

Destination:
CMAC transient data queue

DFHMA083*xx* XML parsing error messages

You can dump the error log to determine the failed program name, transaction, XML exception codes, and other relevant information. Use this information to assist in problem diagnosis.

The following error message is issued because of failed XML parsing or build processing. See Enterprise COBOL for z/OS Programming Guide for any XML exception codes that are reported.

For a sample of XML message structures, see "XML message formats" on page 159.

DFHMA08301E XML-CONVERT-ERRMSG

Explanation

An error was encountered in a CICS Service Flow Runtime XML processing program either when parsing the XML document, the most likely cause, or because of some other unexpected system event. This error condition can be raised in the following CICS SFR programs:

DFHMXMI
DFHMAXMI is called by the CICS SFR interface program DFHMADPL. DFHMAXMI parses the inbound XML request message and returns the DFHMAH header structure in COBOL.

DFHMAXMO
DFHMAXMO is called by the following programs:

- The CICS SFR interface program, DFHMADPL.
- The Navigation Manager, DFHMAMGR.

DFHMAXMO builds the outbound XML reply message using the DFHMAH header structure and any application reply data passed to it.

User response

Use the standard information provided in the message to check for an XML exception code. See the Enterprise COBOL for z/OS Programming Guide for XML exception codes and definitions. Also, examine destination CEEMSG for messages that are produced by Language Environment to determine why the XML parsing failed.

Destination:
CMAC transient data queue.

DFHMA99*xxx* error messages

At run time, the system normally intercepts abends by including an active **EXEC CICS HANDLE ABEND** command. The abend code is determined by issuing the **EXEC CICS ASSIGN** command with the ABCODE option. These abends are reported in the runtime environment with message identifier of DFHMA99999.

DFHMA99999E ABEND-ERRMSG

Explanation:
An abend has occurred in a CICS Service Flow Runtime program, a generated program, or in a target application.

System action:
The service flow processing ends.

User response:
Check the error message details to determine the abend code. Look up the abend code in CICS messages for the cause and take appropriate action.

Destination:
CMAC transient data queue.

DFHMAI*xxxx* postinstallation messages

These messages are issued to the DFHMAINJ job log when CICS SFR verifies the postinstallation procedure.

DFHMAI1000I Validation of input parameters is taking place.

Explanation:
Validation of user-supplied input data is taking place.

System action:
Job continues.

User response:
None.

Destination:
DFHMAINJ job log

DFHMAI1001I No input parameter validation taking place.

Explanation:
No validation of user-supplied input data is taking place, as the NOVALIDATE option has been specified on the invocation of DFHMAINR.

System action:
Job continues.

User response:
None.

Destination:
DFHMAINJ job log

DFHMAI1002I SCIZSAMP customization beginning.

Explanation:
The customization of SCIZSAMP is taking place. Members are copied from the SMP/E SCIZSAMP library to the runtime SCIZSAMP library and user-supplied values applied, as applicable for each member.

System action:
Job continues.

User response:
None.

Destination:
DFHMAINJ job log

DFHMAI1003E Too many characters for *name = value*.

Explanation:
Validation of user-supplied parameter *parmname* has failed because value contains more than the maximum allowable characters.

System action:
Job fails with return code 12.

User response:

Change the value to fit the maximum allowed number of characters and rerun the job.

Destination:
DFHMAINJ job log

DFHMAI1004E Invalid data at position *pos* for *name = value*

Explanation:
Validation of user-supplied parameter *name* has failed because *value* contains invalid data at position *pos* in the variable.

System action:
Job fails with return code 12.

User response:
Replace or remove the invalid data in the value and rerun the job.

Destination:
DFHMAINJ job log

DFHMAI1005E Value not YES or NO for *parmname = value*

Explanation:
Validation of user-supplied parameter *parmname* has failed because value must be YES or NO.

System action:
Job fails with return code 12.

User response:
Change the value to YES or NO and rerun the job.

Destination:
DFHMAINJ job log

DFHMAI1006E Allocate failed for CSDNAME = *csdname*

Explanation:
The CICS DFHCSD file *csdname* cannot be allocated. Check that the specified data set is correct and exists.

System action:
Job fails with return code 12.

User response:
Correct the **CSDNAME** value or create the DFHCSD file before rerunning the job.

Destination:
DFHMAINJ job log

DFHMAI1007E JOB1 does not start with "//+++++ +++ JOB "

Explanation:
Validation of user-supplied parameter **JOB1** has failed because it does not start with "//++++++++ JOB ".

System action:
Job fails with return code 12.

User response:
Ensure that the value of **JOB1** begins with the correct data and rerun the job.

Destination:
DFHMAINJ job log

DFHMAI1008E Invalid JCL continuation for *parmname = value*

Explanation:
Validation of user-supplied parameter *parmname* has failed because of an invalid JCL continuation statement.

System action:
Job fails with return code 12.

User response:
Correct the error and rerun the job.

Destination:
DFHMAINJ job log

DFHMAI1009E Invalid JOB continuation "//" for *parmname = value*

Explanation:
Validation of user-supplied parameter *parmname* has failed because JOB continuation card cannot be only "//"

System action:
Job fails with return code 12

User response:
Correct the error and rerun the job.

Destination:
DFHMAINJ job log

DFHMAI1010I Customizing member: *membername*

Explanation:
Customization has begun for the member *membername*

System action:
Job continues

User response:
None

Destination:
DFHMAINJ job log

DFHMAI1011I SCIZSAMP customization ended without errors.

Explanation:
Customization has completed successfully.

System action:
Job continues.

User response:
None.

Destination:
DFHMAINJ job log

DFHMAI1012I SCIZSAMP customization ended with errors.

Explanation:
Customization has failed. Previous messages describe the errors.

System action:
Job ends.

User response:
Correct the errors found in any previous messages and rerun the job.

Destination:
DFHMAINJ job log

DFHMAI1013E No continuation line found for name.

Explanation:
The continuation character (\) was used in the previous line but the next noncomment line does not have the continuation character as the first nonblank character.

System action:
Job fails.

User response:
Check your input and rerun DFHMAINJ. If the problem still persists, contact IBM.

Destination:
DFHMAINJ job log

DFHMAI1014S *Program* has encountered an unrecoverable error. Novalue error at line *lineno* of *program*: *errtxt*

Explanation:
The REXX exec program, DFHMAINR or DFHMAINX, has encountered an unrecoverable error.

System action:
Job fails.

User response:
Check your input and rerun DFHMAINJ or DFHMAINA. If the problem still persists, contact IBM.

Destination:
DFHMAINJ job log

DFHMAI1015S *Program* has encountered an unrecoverable error. REXX error *rc* at line *lineno* of *program*: *errtxt*

Explanation:
The REXX exec program, DFHMAINX or DFHMAINR, has encountered an unrecoverable error.

System action:
Job fails.

User response:
Check your input and rerun DFHMAINJ or DFHMAINA. If the problem still persists, contact IBM.

Destination:
DFHMAINJ job log

DFHMAI1016S *Program* has encountered an unrecoverable error. Error *err* at line *lineno* of *program*: *errtxt*

Explanation:
The REXX exec program, DFHMAINX or DFHMAINA, has encountered an unrecoverable error.

System action:
Job fails.

User response:
Check your input and rerun DFHMAINJ or DFHMAINA. If the problem still persists, contact IBM.

Destination:
DFHMAINJ job log

DFHMAI1017E Mandatory parameter *parmname* not specified.

Explanation:
A required parameter *parmname* was not supplied.

System action:
Job fails.

User response:
Check your input, add the missing mandatory parameter, and rerun DFHMAINJ.

Destination:
DFHMAINJ job log

DFHMAI1019E Validation failed for *parameter = value*.

Explanation:
Validation of user-supplied parameter *parameter=value* has failed.

System action:
Job fails.

User response:
See following DFHMAI1020E message for an explanation of why the error occurred.

Destination:
DFHMAINJ job log

DFHMAI1020E *Problem data from DFHMAI1019E*.

Explanation:
This message returns the error when data set allocation has failed during validation. The data varies depending on the cause of the error.

System action:
Job fails.

User response:
Correct the reason for the failure and rerun the job. If the problem persists, contact IBM.

Destination:
DFHMAINJ job log

Abends

CICS SFR issues the following abend.

CIAX

While processing in synchronous rollback mode, an error or failure occurred in request processing.

Explanation

System action

The unit of work, in this case the BTS process instance, is not committed. CICS Service Flow Runtime issues an **EXEC CICS ABEND** command with the CANCEL and NODUMP options in the Navigation Manager, DFHMAMGR.

The CICS SFR interface program, DFHMADPL, reports the abend to the CMAC transient data queue. See "DFHMA060xx BTS error messages" on page 136 for information on the run process error message, DFHMA06003.

User response

Check the details of the DFHMA06003 message to determine the problem.

Applying APARs

To apply maintenance or fixes to the runtime environment, use the following process.

About this task

Maintenance is applied using a JCL batch job called DFHMAINA. The JCL is very similar to the DFHMAINJ postinstallation job, in that it contains a subset of the parameters from DFHMAINJ. DFHMAINA invokes a REXX program called DFHMAINX, which looks in member DFHMAINZ for a list of the changed and new members that require maintenance as part of an APAR. DFHMAINZ is updated every time maintenance is applied.

Procedure

1. Edit the DFHMAINA JCL in the SCIZSAMP library, supplying the same parameter values as you do for DFHMAINJ.

   The parameters that you edit are as follows:

   JOB1
   Together with JOB2 and JOB3, JOB1 is used to create the JCL JOB statements for the required jobs in the .SCIZSAMP library. Do not alter the number of + symbols at the beginning of the statement, because these characters are used to substitute the jobname when DFHMAINJ runs.

   JOB2
   JCL JOB statement continued.

   JOB3
   JCL JOB statement continued.

   SHLQ *your.smpe.install.hlq*
   A 1-35 character length value that is the data set name high-level qualifier of the CICS SFR SMP/E installation libraries. This value must match what you specified in the previous step for **syshlq**.

QUAL ***your.runtime.library.hlq***
A 1-35 character length value that is the data set name high-level qualifier of the runtime libraries. This value must match what you specified in the previous step for **hlqual**.

HLQCICS ***your.cics.hlq***
A 1-35 character length value that is the high-level qualifier of the CICS libraries.

HLQCOBOL ***your.cobol.hlq***
A 1-35 character length value that is the high-level qualifier of the COBOL runtime libraries.

HLQCEE ***your.language.environment.hlq***
A 1-35 character length value that is the high-level qualifier of the Language Environment runtime libraries.

WSDIR_REQ ***/your/wsdir/requester/***
The fully qualified name of the Web service pickup directory on zFS that contains the Web service binding file and optionally the WSDL for your Web service requester application. The length of the fully qualified directory name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

CONFIG_REQ ***/your/pipeline/configuration/requester_config.xml***
The name and location of the requester mode pipeline configuration file in zFS.
For example: /usr/lpp/cicsts/samples/pipelines/basicsoap11requester.xml. The pipeline configuration file defines the message handlers that process outbound and inbound Web service requests for your Web service requester application. The length of the fully qualified directory name must not exceed 255 characters and must start with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory and file name are case-sensitive.

SHELF_REQ ***your/shelf/directory/***
The fully qualified name of the directory on zFS that contains subdirectories for the requester mode pipeline configuration files and Web service requester binding files. The length of the fully qualified directory name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

WSDIR_PROV ***/your/wsdir/provider/***
The fully qualified name of the Web service pickup directory on zFS that contains the Web service binding file and optionally the WSDL for your Web service provider application. The length of the fully qualified directory name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

CONFIG_PROV ***your/pipeline/configuration/provider_config.xml***
The name and location of the provider mode pipeline configuration file in zFS.
For example: /usr/lpp/cicsts/samples/pipelines/basicsoap11provider.xml. The pipeline configuration file defines the message handlers that process inbound and outbound Web service requests for your Web service provider application. The length of the fully qualified directory name must not exceed 255 characters and must start with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory and file name are case-sensitive.

SHELF_PROV ***/your/shelf/directory/***
The fully qualified name of the directory on zFS that contains subdirectories for the provider mode pipeline configuration files and Web service provider binding files. The length of the fully qualified name must not exceed 255 characters and must start and end with a /.

Acceptable characters:

```
A-Z a-z 0-9 ./_
```

The directory name is case-sensitive.

PREFIX ***your.prefix***
A 1-7 character length value. The JCL jobname is created as a combination of this value and the name of the member that is customized. For example, if you specify PREFIX CSFR, you will rename every jobname in the members of the runtime SCIZSAMP library with CSFR as the first four characters; for example, //DFHMASET becomes //CSFRASET. If you do not specify a value for this parameter, the sample jobnames are the same as the member names.

Acceptable characters:

```
A-Z a-z 0-9
```

The first character of this value must not be a number.

2. Submit DFHMAINA.

   The job is validated by default. The default is recommended, because any errors in the parameter values are highlighted immediately before the job runs.

   DFHMAINA invokes the REXX program DFHMAINX. This program replaces the SCIZMAC and SCIZLOAD runtime libraries, copies the members listed in DFHMAINZ from the SMP/E SCIZSAMP library to the runtime SCIZSAMP library, and customizes the members using the parameter values that you supplied in DFHMAINA.
3. Check the output for a return code of 0.

   Each member that is in DFHMAINZ has a `Customizing member` message associated with it in the output. You will also see the following messages in the //SYSTSPRT of the DFHMAINA job output:

   ```
   DFHMAI1002I SCIZSAMP customization beginning
   DFHMAI1011I SCIZSAMP customization ended without errors
   ```

   a) If you get a return code of 4 and the following message:

   `CICS SFR SCIZSAMP no members to be customized`

   No updates were made to the SCIZSAMP library. You can continue to the next step.

   b) If you get a return code of 12 and the following message:

   `CICS SFR SCIZSAMP - DFHMAINX could not read member` *xxxxxxxx*

   DFHMAINZ contains a member name that does not exist in the SCIZSAMP library. *xxxxxxxx* is the name of the unrecognized member.

   Read the PTF documentation to check that the member name in question is supposed to be there. You might have to reapply the fix. If the problem still persists, contact IBM.
4. Perform any additional steps as outlined in the APAR documentation.

Results

Chapter 10. Samples

JCL

The following sample JCL is provided with the CICS Service Flow Runtime.

Audit file dump JCL, DFHMABAP

```
//jobname  JOB DFHMABAP,'DFHMABAP',CLASS=A,MSGCLASS=X TYPRUN=SCAN
//*********************************************************************
//* @START_COPYRIGHT@
//*
//* 5655-M15
//*
//* CICS SERVICE FLOW RUNTIME
//*
//* @END_COPYRIGHT@
//*********************************************************************
//*********************************************************************
//* RUN DFHATUP (AUDIT LOG UTILITY PROGRAM)
//*
//*********************************************************************
//ATUP     EXEC PGM=DFHATUP,PARM='N(EN),P(60),T(M)'
//STEPLIB  DD DSN=hlcicsq.SCIZLOAD,DISP=SHR
//*********************************************************************
//*      The output will go to SYSPRINT
//*********************************************************************
//SYSPRINT DD SYSOUT=X,DCB=RECFM=FBA
//AUDITLOG DD DSN=hlq.BTSAUD,
//            SUBSYS=(LOGR,DFHLGCNV),
//            DCB=BLKSIZE=32760
//SYSIN    DD *
PTYPE(processtype)
/*
//
```

BTS repository file dump JCL, DFHMABRP

```
//jobname  JOB DFHMABRP,'DFHMABRP',CLASS=A,MSGCLASS=X TYPRUN=SCAN
//*********************************************************************
//* @START_COPYRIGHT@
//*
//* 5655-M15
//*
//* CICS SERVICE FLOW RUNTIME
//*
//* @END_COPYRIGHT@
//*********************************************************************
//*********************************************************************
//* RUN DFHBARUP (REPOSITORY UTILITY PROGRAM)
//*
//*********************************************************************
//ARUP     EXEC PGM=DFHBARUP,PARM='N(EN),P(60),T(M)'
//STEPLIB  DD DSN=hlcicsq.SCIZLOAD,DISP=SHR
//*********************************************************************
//*      The output will go to SYSPRINT
//*********************************************************************
//SYSPRINT DD SYSOUT=X,DCB=RECFM=FBA
//REPOS    DD DISP=SHR,DSN=hlq.BTS
//SYSIN    DD *
REPOSITORY(REPOS)
/*
//
```

Link3270 Vector Log file dump JCL, DFHMAMVD

The following example shows the JCL that dumps the vector log. Vector logging uses two files, DFHMALVA and DFHMALVB, one of which is always active.

```
//jobname  JOB DFHMAMVD,'DFHMAMVD',CLASS=A,MSGCLASS=H TYPRUN=SCAN
//*********************************************************************
//* @START_COPYRIGHT@
//* VERSION: 0
//*
//* Licensed Materials - Property of IBM
//*
//* 5655-M15
//*
//* (C) Copyright IBM Corp. 2005
//*
//* @END_COPYRIGHT@
//*********************************************************************
//*********************************************************************
//* RUN DFHMAVUP (CICS SFR LINK3270 BRIDGE VECTOR DUMP PROGRAM)
//*
//*********************************************************************
//MAVUP    EXEC PGM=DFHMAVUP
//STEPLIB DD DSN='qual.SCIZLOAD',DISP=SHR
//*********************************************************************
//*     The output will go to SYSPRINT
//*********************************************************************
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//DFHMALVA DD DSN=qual.DFHMALVA,DISP=SHR
//DFHMALVB DD DSN=qual.DFHMALVB,DISP=SHR
//*********************************************************************
//*                                                                   *
//*   SYSIN control card formats                                      *
//*                                                                   *
//**  Card=USERID  *****************************************************
//*                                                                   *
//*  USERID (8 byte CICS userid or '(ALL)'; (ALL) is default.)        *
//*                                                                   *
//*********************************************************************
//SYSIN   DD *
USERID=(ALL)
/*
//
```

Vector file dump

This output is an example of a vector file dump that has been taken after implementing vector logging for a Link3270 server adapter. It includes annotations to explain the important information.

The vector file dump is formatted using the copybook DFHMARLV.

```
   08/04/06                                        CICS SFR Link3270 Navigator Vector Dump   PAGE    1
-------------------------------------------------------------------------------------------------------


*****************************************************************************************************
    Userid:  SFRUSR      Applid:  IYCWZCHV     Tranid:  CMA0     Eibtaskn:   0006551    1
   Request:  MAIVPREQ    Service:                                  Program:  DFHMAVCL
  Proctype:  DFHMAINA                          Process:  SFRUSR000003700336367741718O
      Date:  08/04/06       Time:  14:54       Abstime:  003287832894400       Record seq#:         1
*****************************************************************************************************

   ***  ALLOCATE FACILITY  ***

            Structure:      BRIH
              Version:         1
      Structure length:       180
         Facilitytoken:            (NEW)
               Netname:            (DEFAULT)
              Terminal:            (DEFAULT)
         Transactionid:  CBRA      (Allocate facility)
            Datalength:         0
       Getwaitinterval:   4200000
      Facilitykeeptime:       300
          Facilitylike:            (DEFAULT)
```

```
                    AID:  ENTER     ('   )
              Startcode:  TD        (Terminput)
             Cancelcode:            (NONE)
          ADSDescriptor:         0  (NO)
     Conversationaltask:         1  (YES)
         Cursorposition:         0  (DEFAULT)
```

```
********************************************************************************************************
     Date:  08/04/06       Time:  10:50       Abstime:  003363677428600       Record seq#:
2
********************************************************************************************************


  ***  ALLOCATE FACILITY RESPONSE
***


            Structure:
BRIH
              Version:         2
(Extended)
     Structure length:
180
        Facilitytoken:
X'0094000100000001'
            Region ID:
ZCHV
              Netname:
AAA}
             Terminal:
AAA}
        Transactionid:  CBRA      (Allocate
facility)
           Datalength:
0
          Return code:         0
(OK                     )
            Comp code:
0
          Reason code:
0
             Function:
```

```
********************************************************************************************************
     Date:  08/04/06       Time:  10:50       Abstime:  003363677428600       Record seq#:
3
********************************************************************************************************


  ***  INBOUND
***



            Structure:      BRIH
 2
              Version:         2
(Extended)
     Structure length:
180
        Facilitytoken:
X'0094000100000001'
            Region ID:
ZCHV
              Netname:
AAA}
             Terminal:
AAA}
        Transactionid:
CMAV
           Datalength:
180
      Getwaitinterval:   4200000                                                          3
     Facilitykeeptime:
300
         Facilitylike:
(DEFAULT)
                    AID:  ENTER
('   )
```

```
            Startcode:  TD
(Terminput)
           Cancelcode:
(NONE)
       ADSDescriptor:         0
(NO)
   Conversationaltask:         1
(YES)
      Cursorposition:
5
```

```
********************************************************************************************************
     Date:  08/04/06       Time:  10:50       Abstime:  003363677428730       Record seq#:
4
********************************************************************************************************


  ***  OUTBOUND SEND MAP  (1804)
***


            Structure:
BRIH
              Version:         2
(Extended)
     Structure length:
180
       Facilitytoken:
X'0094000100000001'
            Region ID:
ZCHV
              Netname:
AAA}
             Terminal:
AAA}
         Transactionid:
CMAV
            Datalength:
1156
          Return code:         0  (OK                       )
 4
            Comp code:
0
          Reason code:
0
             Function:
            Abendcode:            (NONE)
        Taskendstatus:     65536
(Endtask)
  Remainingdatalength:
0
  08/04/06                                        CICS SFR Link3270 Navigator Vector
Dump
 ---------------------------------------------------------------------------------------------------


                SYSID:
ZCHV
    Nexttransactionid:
CMAV
     Nexttranidsource:         0
(Normal)
          Erroroffset:
0
                Seqno:
1


   BRIV vector length:       976
 5
   SM Erase indicator:
ERASE
          SM ERASEAUP:
NO
     SM Free Keyboard:
NO
             SM Alarm:
NO
             SM FRSET:
NO
              SM Last:
```

```
NO
    SM Wait indicator:
NO
   SM cursor position:        -1
(DYNAMIC)
          SM MSR data:
NONE
            SM Mapset:
DFHMAB1
               SM Map:
MAPA
    SM data indicator:
DEFAULT
       SM data length:
882
       SM data offset:
92
       SM ADSD length:
0
       SM ADSD offset:
0
             SM ACCUM:
NO
```

```
   (partial ADS from dump)                                                    6

           0 :  | 00000000 00000000 00000000 0000F000 00005C5C 5C40C3E4 |    | ..............0...*** CU
|
          24 :  | E2E3D6D4 C5D940C9 C4C5D5E3 C9C6C9C3 C1E3C9D6 D5405C5C |    | STOMER IDENTIFICATION **
|
          48 :  | 5C0000F0 000000C3 C9C6E2F0 F1404040 400000F0 00000000 |    | *..0...CIFS01    ..0....
|
          72 :  | 00000000 00000000 000000C8 00F40040 40404040 00006000 |    | ...........H.4.     ..-. |
```

1. This section is the header for the dump file and includes the following fields:

Field name	Description
Userid	The user ID under which the vector log file records were written and service flow processing was running.
Applid	The APPLID of the CICS region where the Link3270 server adapter was running.
Eibtaskn	The task number assigned by CICS to the Link3270 server adapter transaction.
Request	The name of the request that was being processed. This field corresponds to the request name of the service flow.
Service	The service name used in the Link3270 server adapter. This field can be blank. If specified, it is used to provide part of the Link3270 state file key when multiple bridge facilities are allocated, and remain in use for a specific user ID in adapter server processing.
Program	The name of the Link3270 server adapter.
Proctype	The BTS process type of the request or the process under which the Link3270 server adapter was running.
Process	The process name of the request processing instance under which the Link3270 server adapter was running.
Date	The date that the records were written to the vector log file.
Time	The time that the records were written to the vector log file.
Abstime	The time as reported by CICS when the vector log file records were written. This time is retrieved using an **EXEC CICS ASKTIME** command.
Record seq#	The sequential count of the specific record written to the vector log file by the Link3270 server adapter.

2. The fields `Structure` down to `Datalength` are common to both inbound and outbound messages.
3. The fields `Getwaitinterval` down to `Cursorposition` are used only for inbound messages.
4. The field `Return code` and onwards refer to inbound or outbound response messages only.
5. **`SEND MAP`** vector. The format is different for inbound and outbound vectors. Only one appears for every record. Each inbound and outbound vector message structure, BRIV, is preceded by the Link3270 bridge header structure BRIH.
6. The beginning of the application data structure (ADS) or screen buffer. This data is provided when you run full vector logging. If you have selected vector logging trace, you do not get this information. When using full vector logging, the vector data that is displayed as contents in the vector dump is the accumulated map buffer, text data, or screen buffer that is returned to the generated Link3270 navigator. The buffer contents in the vector dump might not always contain the individual vector data returned from the Link3270 bridge.

Conversion template for DFHMADPL

You might have to add a conversion template, depending on how the service requester is invoking a service flow.

If you are using standard CICS conversion, you can add this template as required to the CICS conversion table, DFHCNV, for code page conversion. The conversion template must also be present in the custom CICS Service Flow Runtime conversion table to support the CICS Web Interface. Define the conversion template in the following macros:

- DFHCNV TYPE=INITIAL (defines the beginning of the conversion table)
- DFHCNV TYPE=FINAL (defines the end of the conversion table)

Use the following conversion template to create a load module in the required CICS load library. The asterisks are required in column 72 as continuation characters.

```
DFHCNV TYPE=INITIAL
DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=DFHMADPL,USREXIT=NO,      *
     SRVERCP=037 1 ,CLINTCP=8859-1  2
DFHCNV TYPE=SELECT,OPTION=COMPARE,OFFSET=0,DATA='<?XM'
DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,              *
               DATALEN=32760,LAST=YES
DFHCNV TYPE=SELECT,OPTION=COMPARE,OFFSET=0,DATA='<?xm'
DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,              *
               DATALEN=32760,LAST=YES
DFHCNV TYPE=SELECT,OPTION=COMPARE,OFFSET=0,DATA='MAH '
DFHCNV TYPE=FIELD,OFFSET=00,DATATYP=CHARACTER,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=04,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=08,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=12,DATATYP=CHARACTER,DATALEN=16
DFHCNV TYPE=FIELD,OFFSET=28,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=32,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=36,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=40,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=44,DATATYP=CHARACTER,DATALEN=16
DFHCNV TYPE=FIELD,OFFSET=60,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=64,DATATYP=CHARACTER,DATALEN=52
DFHCNV TYPE=FIELD,OFFSET=116,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=120,DATATYP=CHARACTER,DATALEN=232
DFHCNV TYPE=FIELD,OFFSET=352,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=356,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=360,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=364,DATATYP=CHARACTER,DATALEN=20
DFHCNV TYPE=FIELD,OFFSET=384,DATATYP=CHARACTER,DATALEN=32376, *
     LAST=YES
```

1. The SRVERCP keyword on the DFHCNV TYPE=ENTRY macro determines the server code page in which character data associated with the specified resource is encoded in the z/OS server. Such data is assumed to be encoded in EBCDIC.
2. The CLINTCP keyword on the DFHCNV TYPE=ENTRY macro determines the default client code page in which the character data associated with the specified resource is encoded when it is received by or sent from the z/OS server. In general, such data is assumed to be encoded in ASCII. However, the data

might be encoded in EBCDIC; for example, for data passed through the CICS Web Server Plug-in. In this case, the client and server code pages are likely to be different, even though both are EBCDIC.

XML message formats

The following samples are of the CICS Service Flow Runtime message structure in XML.

XSD for request message entirely in XML

```
<?xml version="1.0"?>
<schema
    targetNamespace="http://www.DFHMAXMI.com/schemas/DFHMAXMIInterface"
    xmlns="http://www.w3.org/2001/XMLSchema" xmlns:cbl="http://www.DFHMAXMI.com/schemas/
DFHMAXMIInterface">
    <complexType name="DFHMAMSG">
        <sequence>
            <element name="dfhmah" type="cbl:dfhmamsg_dfhmah"/>
            <element name="dfhmah2" type="cbl:dfhmamsg_dfhmah2"/>
            <element name="dfhmaad">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="24576"/>
                    </restriction>
                </simpleType>
            </element>
        </sequence>
    </complexType>
    <group name="dfhmamsg_dfhmah">
        <sequence>
            <element name="dfhmah__strucid">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="4"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__version">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__struclength">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__userid">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
```

```
                <maxLength value="8"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__format">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="8"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__returncode">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__compcode">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__mode">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__suspstatus">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__abendcode">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="4"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__message">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="12"/>
            </restriction>
        </simpleType>
    </element>
```

```
            <element name="dfhmah__uowcontrol">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__processtype">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="8"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__processname">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="36"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__requestname">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="8"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__datalength">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__failed__procname">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="36"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__failed__proctype">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
```

```
                <maxLength value="8"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__failed__tranid">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="4"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__replytoq">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="48"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__replytoqmgr">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="48"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__msgid">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="24"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__correlid">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="24"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__failed__program">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
```

```
                    <maxLength value="8"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__failed__node">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="32"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__linktype">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__more__data__ind">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__bridge__rc">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__statetoken">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="16"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__reserved2">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="4"/>
                </restriction>
            </simpleType>
        </element>
    </sequence>
</group>
<complexType name="dfhmamsg_dfhmah">
    <group ref="cbl:dfhmamsg_dfhmah"/>
</complexType>
<group name="dfhmamsg_dfhmah2">
    <sequence>
        <element name="dfhmah2__strucid">
```

```
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="4"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__version">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__struclength">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__reserved">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="8"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__format">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="8"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__datalength">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__transid">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="4"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__receive__type">
```

```
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__next__transid">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="4"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__use__fkeeptime__ind">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__facilitykeeptime">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__facilitylike">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="4"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__getwaitinterval">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__vector__logging">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__deallocate__ind">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah2__send__aid__first">
```

```
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__initial__aid__byte">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__clientip__address">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="39"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__resptime">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__applresptime">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__xml__programid"
type="cbl:dfhmamsg_dfhmah2_dfhmah2__xml__programid"/>
            <element name="dfhmah2__reserved2">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="36"/>
                    </restriction>
                </simpleType>
            </element>
        </sequence>
    </group>
    <complexType name="dfhmamsg_dfhmah2">
        <group ref="cbl:dfhmamsg_dfhmah2"/>
    </complexType>
    <group name="dfhmamsg_dfhmah2_dfhmah2__xml__programid">
        <sequence>
            <element name="dfhmah2__xml__program">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
```

```
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="7"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__xml__program__tag">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
        </sequence>
    </group>
    <complexType name="dfhmamsg_dfhmah2_dfhmah2__xml__programid">
        <group ref="cbl:dfhmamsg_dfhmah2_dfhmah2__xml__programid"/>
    </complexType>
    <element name="dfhmamsg" type="cbl:DFHMAMSG"/>
</schema>
```

```
<?xml version="1.0"?>
<schema
    targetNamespace="http://www.DFHMAXMO.com/schemas/DFHMAXMOInterface"
    xmlns="http://www.w3.org/2001/XMLSchema" xmlns:cbl="http://www.DFHMAXMO.com/schemas/
DFHMAXMOInterface">
    <complexType name="DFHMAMSG">
        <sequence>
            <element name="dfhmah" type="cbl:dfhmamsg_dfhmah"/>
            <element name="dfhmah2" type="cbl:dfhmamsg_dfhmah2"/>
        </sequence>
    </complexType>
    <group name="dfhmamsg_dfhmah">
        <sequence>
            <element name="dfhmah__strucid">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="4"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__version">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__struclength">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah__userid">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
```

```
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="8"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__format">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="8"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__returncode">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__compcode">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__mode">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__suspstatus">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__abendcode">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="4"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah__message">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
```

```
                <restriction base="string">
                    <maxLength value="12"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__uowcontrol">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__processtype">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="8"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__processname">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="36"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__requestname">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="8"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__datalength">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__failed__procname">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="36"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__failed__proctype">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
```

```
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="8"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__failed__tranid">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="4"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__replytoq">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="48"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__replytoqmgr">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="48"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__msgid">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="24"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__correlid">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="24"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__failed__program">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
```

```
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="8"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__failed__node">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="32"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__linktype">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__more__data__ind">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__bridge__rc">
            <simpleType>
                <restriction base="int">
                    <minInclusive value="-999999999"/>
                    <maxInclusive value="999999999"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__statetoken">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="16"/>
                </restriction>
            </simpleType>
        </element>
        <element name="dfhmah__reserved2">
            <annotation>
                <appinfo source="http://www.wsadie.com/appinfo">
                    <initialValue kind="SPACE"/>
                </appinfo>
            </annotation>
            <simpleType>
                <restriction base="string">
                    <maxLength value="4"/>
                </restriction>
            </simpleType>
        </element>
    </sequence>
</group>
<complexType name="dfhmamsg_dfhmah">
```

```
        <group ref="cbl:dfhmamsg_dfhmah"/>
    </complexType>
    <group name="dfhmamsg_dfhmah2">
        <sequence>
            <element name="dfhmah2__strucid">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="4"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__version">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__struclength">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__reserved">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="8"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__format">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="8"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__datalength">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__transid">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
```

```
                <maxLength value="4"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__receive__type">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__next__transid">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="4"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__use__fkeeptime__ind">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__facilitykeeptime">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__facilitylike">
        <annotation>
            <appinfo source="http://www.wsadie.com/appinfo">
                <initialValue kind="SPACE"/>
            </appinfo>
        </annotation>
        <simpleType>
            <restriction base="string">
                <maxLength value="4"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__getwaitinterval">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__vector__logging">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
                <maxInclusive value="999999999"/>
            </restriction>
        </simpleType>
    </element>
    <element name="dfhmah2__deallocate__ind">
        <simpleType>
            <restriction base="int">
                <minInclusive value="-999999999"/>
```

```
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__send__aid__first">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__initial__aid__byte">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__clientip__address">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="39"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__resptime">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__applresptime">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="-999999999"/>
                        <maxInclusive value="999999999"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__xml__programid"
type="cbl:dfhmamsg_dfhmah2_dfhmah2__xml__programid"/>
            <element name="dfhmah2__reserved2">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="36"/>
                    </restriction>
                </simpleType>
            </element>
        </sequence>
    </group>
    <complexType name="dfhmamsg_dfhmah2">
        <group ref="cbl:dfhmamsg_dfhmah2"/>
    </complexType>
    <group name="dfhmamsg_dfhmah2_dfhmah2__xml__programid">
```

```
        <sequence>
            <element name="dfhmah2__xml__program">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="7"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="dfhmah2__xml__program__tag">
                <annotation>
                    <appinfo source="http://www.wsadie.com/appinfo">
                        <initialValue kind="SPACE"/>
                    </appinfo>
                </annotation>
                <simpleType>
                    <restriction base="string">
                        <maxLength value="1"/>
                    </restriction>
                </simpleType>
            </element>
        </sequence>
    </group>
    <complexType name="dfhmamsg_dfhmah2_dfhmah2__xml__programid">
        <group ref="cbl:dfhmamsg_dfhmah2_dfhmah2__xml__programid"/>
    </complexType>
    <element name="dfhmamsg" type="cbl:DFHMAMSG"/>
</schema>
```

Sample request message entirely in XML

The following sample is a request message in XML.

For a description of the fields in the CICS Service Flow Runtime message header, DFHMAH, see “Request message headers” on page 47.

The value specified in the **dfhmah_datalength** element represents the length of the application data in flat format; not the length of the XML elements between the start application data tag, **<dfhmaad>**, and the end application data tag, **</dfhmaad>**.

```
<?xml version="1.0" encoding="UTF-8"?>
          <cbl:dfhmamsg>
           <dfhmah>
            <dfhmah__strucid>MAH</dfhmah__strucid>
            <dfhmah__version>2</dfhmah__version>
            <dfhmah__struclength>384</dfhmah__struclength>
            <dfhmah__userid>USER0001</dfhmah__userid>
            <dfhmah__format> </dfhmah__format>
            <dfhmah__returncode>0</dfhmah__returncode>
            <dfhmah__compcode>0</dfhmah__compcode>
            <dfhmah__mode>0</dfhmah__mode>
            <dfhmah__suspstatus>0</dfhmah__suspstatus>
            <dfhmah__abendcode>NONE</dfhmah__abendcode>
            <dfhmah__message>message one</dfhmah__message>
            <dfhmah__uowcontrol>0</dfhmah__uowcontrol>
            <dfhmah__processtype>DFHMAINA</dfhmah__processtype>
            <dfhmah__processname></dfhmah__processname>
            <dfhmah__requestname>MAIVPREQ</dfhmah__requestname>
            <dfhmah__datalength>22</dfhmah__datalength>
            <dfhmah__failed__procname> </dfhmah__failed__procname>
            <dfhmah__failed__proctype> </dfhmah__failed__proctype>
            <dfhmah__failed__tranid> </dfhmah__failed__tranid>
            <dfhmah__replytoq> </dfhmah__replytoq>
            <dfhmah__replytoqmgr> </dfhmah__replytoqmgr>
            <dfhmah__msgid> </dfhmah__msgid>
            <dfhmah__correlid> </dfhmah__correlid>
            <dfhmah__failed__program> </dfhmah__failed__program>
            <dfhmah__failed__node> </dfhmah__failed__node>
            <dfhmah__linktype></dfhmah__linktype>
            <dfhmah__more__data__ind>0</dfhmah__more__data__ind>
```

```
              <dfhmah__bridge__rc>0</dfhmah__bridge__rc>
              <dfhmah__statetoken> </dfhmah__statetoken>
              <dfhmah__reserved2> </dfhmah__reserved2>
             </dfhmah>
             <dfhmaad>
               <cif__input>
                 <cifflag>D</cifflag>
                 <account__no>10000</account__no>
                 <user__id></user__id>
                 <user__password></user__password>
               </cif__input>
             </dfhmaad>
            </cbl:dfhmamsg>
```

Chapter 11. Supplementary information

Reference information for CICS Service Flow Runtime.

Server runtime programs

The following table lists all of the programs and describes the processing performed by each.

Table 22. CICS Service Flow Runtime programs

Program name	Description	Transaction ID	Details
DFHMADPL	CICS SFR interface program	User defined	This program defines and runs the BTS process that implements an instance of the service flow. It runs under the transaction ID of the invoking program.
DFHMAF	FEPI 3270 data stream conversion	User defined	The FEPI 3270 data stream conversion program is used to convert 3270 data streams to a screen. It is also used to build 3270 data streams from the screen. It is a subprogram that runs under the transaction ID of the FEPI server adapter. It is used in FEPI server adapter processing only. It must be included on the LINKEDIT step when compiling FEPI server adapters.
DFHMALFC	Link3270 VSAM state cleanup	CMAF	This program monitors the Link3270 State VSAM file DFHMAL2F, and deletes expired Link3270 facility session state data. It also deallocates associated Link3270 bridge facilities that CICS has not automatically deleted because the facility is inactive for the keeptime interval. It is used for complex service flows and persistent simple flows that include a Link3270 server adapter.
DFHMALFD	Link3270 facility deallocation cleanup	CMAD	The Link3270 Facility Deallocate Cleanup program is used to deallocate existing bridge facilities and delete the associated facility business state data, whether that data is stored in a temporary storage queue or VSAM file.
DFHMALIN	Link3270 initiation	CMAI	The Link3270 initiation program retrieves any target CICS application transaction COMMAREA information and any TCTUA information from the CICS region where the Link3270 bridge facility is currently allocated, and populates that same information in a second CICS region, before running the next target CICS application transaction routed to that second CICS region.
DFHMALNM	Link3270 maintenance	User defined	The Link3270 maintenance program is invoked by the Link3270 server adapter during request processing. It starts and ends processing for Link3270 server adapters. It is a subprogram that runs under the transaction ID of the Link3270 server adapter.

Table 22. CICS Service Flow Runtime programs (continued)

Program name	Description	Transaction ID	Details
DFHMALSC	Link3270 TSQ state cleanup	CMAK	The TSQ state cleanup program browses the Link3270 state temporary storage queues and deletes expired Link3270 facility session state data. It also de-allocates associated Link3270 bridge facilities that CICS has not automatically deleted, because the facility is inactive for the keeptime interval. It is used for Link3270 service flows that are simple and not persistent.
DFHMALTS	Link3270 TSQ state management	User defined	The TSQ state management program saves, retrieves, and deletes state information from the Link3270 State TSQ for Link3270 server adapters in a simple service flow that are not persistent. It runs under the transaction ID of the Link3270 server adapter.
DFHMAMGR	Navigation Manager	CMAM	The Navigation Manager runs as DFHROOT in all BTS processes. The functions it performs depend on the deployment pattern of the service flow: • When the service flow is complex, the Navigation Manager invokes a flow navigator to run the service flow as defined in the service flow repository file. • When the service flow is simple, for example simple screen sequencing, the Navigation Manager invokes the server adapters necessary to run the service flow, as defined in the service flow properties file.
DFHMAVCL	Link3270 vector logging	User defined	The Link3270 vector logging program writes Link3270 vector data to the vector log files DFHMALVA and DFHMALVB. It is a subprogram that runs under the transaction ID of the Link3270 server adapter.
DFHMAVCP	Link3270 vector processing	User defined	The Link3270 vector processing program is invoked by the Link3270 server adapter during request processing. This program sends vectors to and receives vectors from the Link3270 bridge program DFHL3270 to interface with a CICS target application, where a single send and receive structure inclusive of the Link3270 bridge header and appropriate input or output vector header is not greater than 32 000 bytes. It is a subprogram that runs under the transaction ID of the Link3270 server adapter.
DFHMAXMI	XML header to COBOL converter	User defined	This program is called by the CICS SFR interface program, DFHMADPL when the request message header is in XML format. The XML header to COBOL converter program converts the XML into a COBOL data structure so that it can be processed by the CICS Service Flow Runtime.

Table 22. CICS Service Flow Runtime programs (continued)			
Program name	**Description**	**Transaction ID**	**Details**
DFHMAXMO	COBOL to XML converter	User defined	This program is called by theCICS SFR interface program, DFHMADPL when the service requester expects a response in XML format. The COBOL-to-XML converter program converts the COBOL data structure into XML so that it can be processed by the service requester.
User defined	Flow navigator	User defined	A child activity of the Navigation Manager and parent activity to the server adapters. Flow navigators perform request processing by invoking the required server adapters in the correct order, and manages the state of each server adapter. The flow navigator program is generated from service flow project tools for complex service flows.

CICS Service Flow Runtime uses the following IBM WebSphere MQ programs and transactions when the service requester uses WebSphere MQ to invoke a service flow.

These components are part of the WebSphere MQ-CICS bridge. WebSphere MQ-CICS bridge is not part of CICS Service Flow Runtime, but it must be used as the interface between a service requester using WebSphere MQ and the CICS Service Flow Runtime. WebSphere MQ-CICS bridge enables an application, not running in a CICS environment, to run a program or transaction in CICS and receive a response.

Table 23. WebSphere MQ-CICS bridge programs used by CICS Service Flow Runtime			
Program name	**Description**	**Transaction ID**	**Details**
DFHMQBP0	WebSphere MQ-CICS bridge DPL program	CKBP	This program pulls the message off the request queue and links to the CICS SFR interface program with the DFHMAH header information and the application data.
DFHMQBR0	WebSphere MQ-CICS bridge monitor program	CKBR	This program monitors the request queue for messages. When a message arrives in the queue, the bridge monitor task starts the WebSphere MQ CICS bridge link program.

Server runtime files

The following table lists the files that CICS Service Flow Runtime uses during request processing.

Every file is a Key Sequenced Data Set (KSDS).

Table 24. CICS Service Flow Runtime VSAM files		
File name	**Summary**	**Description**
DFHMAASF	Service flow repository	This file contains the definitions for every service flow that is deployed in the CICS region.
DFHMACOF	FEPI (SLU) connection	This file is used by FEPI server adapters. It contains the FEPI nodes that are allocated to avoid allocating the same node twice.

Table 24. CICS Service Flow Runtime VSAM files (continued)		
File name	**Summary**	**Description**
DFHMAC1F	FEPI (SLU) alternate connection	This file is used to retrieve or establish active FEPI connections and conversations using the signed-on user ID and the FEPI pool name as defined in the FEPI server adapter properties.
DFHMAL2F	Link3270 business state	This file is used by Link3270 server adapters. It contains Link3270 facility business (application) state information for the allocated facility. For example; the last BMS ADS sent and the vector header information.
DFHMATIF	FEPI target interaction	This file is used by FEPI server adapters. It contains the last screen buffer that was sent or received using the allocated conversation for the target node.
DFHMALVA	Vector log	This file is the first of two files that is used by Link3270 server adapters to log Link3270 bridge vector information that is sent and received by the CICS target application.
DFHMALVB	Vector log	This file is the second of two files that is used by Link3270 server adapters to log Link3270 bridge vector information that is sent and received by the CICS target application.

Transactions supplied with CICS SFR

The following table lists all of the transactions that are supplied with CICS Service Flow Runtime.

Transaction id	Program	Security category	Description
CMAD	DFHMALFD	2	Runs the Link3270 cleanup program to deallocate existing bridge facilities and delete business state data.
CMAF	DFHMALFC	2	Runs the Link3270 state cleanup program for Link3270 server adapters that use VSAM files.
CMAK	DFHMALSC	2	Runs the Link3270 state cleanup program for Link3270 server adapters that use temporary storage queues.
CMAM	DFHMAMGR	2	Runs the Navigation Manager.
CMAN	DFHMAMUI	2	Flow management transaction.
CMAO	DFHMASWS	2	Runs the Web service server adapter.
CMAS	DFHMASDP	2	Runs the program link server adapter.
CMAU	DFHMASCQ	2	Runs the queue server adapter.
CMIT	DFHMAINS	2	Installs service flows when you restart CICS.

Chapter 12. Glossary

This glossary contains terms relevant to both CICS Service Flow Runtime and service flow project tools.

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | #

A

access pattern
See deployment pattern.

activity
One part of a process managed by CICS business transaction services. Activities implement the business logic. Typically, an activity is part of a business transaction and is executed by a normal CICS transaction responding to CICS BTS events.

asynchronous
Pertaining to events that are not synchronized in time or do not occur in regular or predictable time intervals.

asynchronous processing
A means of distributing the processing of an application between systems in an intercommunication environment. The processing in each system is independent of the session on which requests are sent and replies are received. No direct correlation can be made between requests and replies and no assumptions can be made about the timing of replies.

auditing
Collecting and recording information about the state of the CICS Service Flow Runtime for the purpose of diagnostics and tracing. CICS Service Flow Runtime uses BTS facilities for auditing.

authentication
In computer security, verification of the identity of a user or process and the construction of a data structure that contains the privileges that were granted to the user or process.

B

basic mapping support (BMS)
An interface between CICS and application programs that formats input and output display data and routes multiple-page output messages without regard for control characters used by various terminals.

business process
A group of logically related activities that use the resources of the organization to provide defined results in support of the organization's objectives.

business transaction
A self-contained business function; for example, the booking of an airline ticket. Traditionally, in CICS a business transaction might be implemented as many user transactions; the booking of the airline ticket might be undertaken by transactions that inquire about availability, reserve the seat, deal with payment, and print the ticket. Using BTS, a business transaction might be implemented as multiple activities.

C

CICS Business Transaction Services (CICS BTS)
A CICS domain that supports an application programming interface (API) and services that simplify the development of business transactions. Using BTS, each action that comprises the business transaction is implemented as one or more CICS transactions.

communication area (COMMAREA)
A CICS area that is used to pass data between tasks that communicate with a given terminal. The area can also be used to pass data between programs within a task.

compensation
The act of modifying the effects of a completed activity. How this is implemented is decided by the application designer, but often means the undoing, or reversing, of the actions that the activity took.

connector
A connector is a well-defined, durable communication or programming interface to an Enterprise Information System. A connector provides a means of accepting data in a definable format, invoking an operation, and receiving results in a definable format.

D

data-container
A named area of storage, maintained by BTS, and used to pass data between activities, or between different invocations of the same activity. Each data container is associated with an activity; it is identified by its name and by the activity for which it is a container. An activity can have any number of containers, if they all have different names.

data conversion
The process of changing data from one form of representation to another.

deployment pattern
Sometimes referred to as an *access pattern*, a deployment pattern refers to the way in which a service flow complies with a set of well-defined usage patterns for processing in a supported runtime environment.

distributed program link (DPL)
A function of CICS intersystem communication that enables an application program to ship LINK requests to another application program on a different instance of CICS.

E

enterprise information system (EIS)
The applications that comprise an enterprise's existing system for handling company-wide information. An enterprise information system offers a well-defined set of services that are exposed as local or remote interfaces or both.

enterprise information system interface
Represents the data source in an enterprise information system; for example 5250 and 3270 screens, COBOL record descriptions, and transactions. Using service flow project tools, developers can model and compose these interfaces into a more SOA-compliant programmatic interface, enabling the enterprise to transform or adapt to a new set of operations and methods that move the application towards a service oriented architecture.

F

FEPI
See front end programming interface.

flow
See service flow.

flow navigator
A program that performs server adapter processing, managing states during the microflow processing and invoking server adapters in the correct order. The flow navigator is generated by service flow project tools.

front end programming interface (FEPI)
A separately installable function of CICS that enables CICS programs to interact with other 3270-based applications through virtual terminal sessions.

H

host
A computer system, such as a mainframe transaction processing system, on which EIS applications reside.

host application
An application residing on the host computer system.

I

importer
A component in the service flow project tools that is used to populate parts of the information model from existing resources. These resources can be data format definitions for messages or control blocks used by host applications, screen format definitions, for either entire screens or parts of screens, existing navigation information such as emulator macros, or captures of actual screens.

inline DPL
The use of a DPL command within a FEPI service flow that is modeled using the service flow project tools. This feature allows the generation of a FEPI server adapter that not only performs 3270 screen navigation, but also can connect to a back-end program for data access and processing using a distributed program link. See distributed program link.

interface
The contract between the service requester and the service provider expressed as a defined set of operations and the defined message formats for each operation. An Interface component describes sequences of messages that a service sends or receives or both. It does so by grouping related messages into operations. An operation is a sequence of input and output messages, and an interface is a set of operations. Thus, an interface defines the design of the application.

J

journal
A set of one or more data sets to which records are written during a CICS run.

journaling
The recording of information onto any journal (including the system log), for possible subsequent processing by the user.

Link3270 bridge
A facility in CICS that provides a simplified interface using LINK, ECI, and EXCI. An application uses the Link3270 bridge to run 3270 transactions by linking to the DFHL3270 program in the router region and passing a COMMAREA that identifies the transaction to be run and contains the data used by the user application. If the target application used BMS, the reply is presented in the form of an application data structure (ADS), another name for the symbolic map that is generated by the BMS macros used to define the mapping of the 3270 terminal screen.

M

message
A set of data that is passed from one application to another. A message can be modeled by a message definition, which describes the structure and content of the message. Messages must have a structure and format that is agreed by the sending and receiving applications.

O

operation
A service that can be requested from an object to effect behavior. A Web service can have multiple operations. An operation has a signature, which might restrict the actual parameters that are possible. EIS operations are generally not independent of each other.

P

persistence
An instance state of data that is maintained across session boundaries or of an object that continues to exist after the execution of the program or process that created it, usually in nonvolatile storage such as a database system.

process
In BTS, a collection of one or more activities. A process is the largest unit with which CICS BTS can work and has a unique name by which it can be referenced and invoked.

process container
A data-container associated with a process. Process containers can be read by all the activities that make up the process. Note that they are not the same as the root activity's containers.

R

request message
A message sent by the service requester to invoke a service flow to process a business transaction.

Resource Access Control Facility (RACF)
An IBM licensed program that provides access control by identifying users to the system, verifying users of the system, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected resources. RACF is included in z/OS Security Server and is also available as a separate program for the MVS and VM environments. In CICS Service Flow Runtime, RACF is used to make sure that a user has the authority to run a particular CICS DPL bridge task.

root activity
The activity at the top of the activity tree; it has no parent activity. The root activity is the control program for a business transaction that represents the start and the end of the process. It initiates and controls a set of child activities.

runtime environment
The supported environment to which a service flow can be deployed.

run time
The time period during which an instance of a service flow is operational, where a business transaction is processed and completed in the CICS region.

S

screen
In its native state, a screen represents the user interface to a 3270 or 5250 application on a host system. A single host application can contain many screens, each of which has a purpose in the context of the application. Screens contain both text and control (or formatting functions) and traditionally display as “green screens” on 3270 or 5250 terminals.

screen navigation
A form of data transfer between two application programs in which the first program accesses the second program through a terminal emulator or other communications program, and obtains data that appears at known screen locations if the second program was being accessed by a human operator.

server adapter
A program that is invoked to perform the business transaction activity defined in a service flow at build time.

service flow
A graphical representation of the sequence of activities that are performed in accordance with the business processes of an enterprise. A service flow consists of a graph of nodes, with defined entry and exit points where each node represents invoking a service operation, controlling the flow of the sequence, or performing reusable business logic.

Service Flow Modeler
An eclipse-based application integration toolset that enables users to expose their existing applications as a service-like interface, facilitating the move to Service Oriented Architecture (SOA). Service Flow Modeler is available in the IBM Developer for Z product.

service provider
The collection of software that provides a Web service.

service provider application
An application that is used in a service provider. Typically, a service provider provides the business logic components of a service provider.

service requester
The collection of software that requests a Web service from a service provider.

service requester application
An application that is used in a service requester. Typically, a service requestor application provides the business logic component of a service requester.

synchronous
Relating to two or more processes that depend upon the occurrences of specific events, such as a common timing signal.

T

target application
An application that contains the data or information required to fulfill the business transaction.

transaction
A unit of processing consisting of one or more application programs, affecting one or more objects, that is initiated by a single request.

transform
To change the structure and values of data from one form to another. At build time, a developer can use service flow project tools to transform existing interfaces in an EIS to facilitate participation of EIS applications in an SOA.

W

Web service
A software system that supports interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format; specifically, Web Service Description Language, or WSDL.

Web service description
An XML document by which a service provider communicates the specifications for invoking a Web service to a service requester. Web service descriptions are written in Web Service Description Language (WSDL).

Web Services Definition Language (WSDL)
An XML application for describing Web services. It separates the descriptions of the abstract functions offered by a service and the concrete details of a service, such as how and where that function is offered.

#

3270 data stream
The commands, control codes, orders, attributes, and data or structured fields for 3270 devices that are transmitted between an application program and a terminal. Data being transferred from or to an allocated primary or tertiary device, or to the host system, as a continuous stream of data and 3270 Information Display System control elements in character form.

Notices

This information was developed for products and services offered in the United States of America. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Client Relationship Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

IBM CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 5 (CICS TS 5.5) are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- CICS security
- Developing for external interfaces
- Reference: application development
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS TS 5.5, but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS TS 5.5 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services

- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- CICS Transactions
- CICSPlex® System Manager (CICSPlex SM) Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS TS 5.5, but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

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

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the United States and other countries.

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

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

The Stack Exchange name and logos are trademarks of Stack Exchange Inc.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights
Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information (PII) is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect PII. If this Software Offering uses cookies to collect PII, specific information about this offering's use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other PII for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other PII for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that do not collect PII. These cookies cannot be disabled.

For CICS Explorer®:

Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user's user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect PII from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled *Cookies, Web Beacons and Other Technologies* and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

F

I

J

L

M

P

Q

R

S

T

U

V

W

X