CICS Service Flow Runtime

User's Guide

Version 3 Release 1

SC34-5899-10

CICS Service Flow Runtime

User's Guide

Version 3 Release 1

SC34-5899-10

Note: Before using this information and the product it supports, read the information in “Notices” on page 393.

Eleventh edition (July 2010)

This edition applies to Version 3 Release 1 of CICS Service Flow Runtime (product number 5655-M15) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can make comments on this information via e-mail at idrcf@hursley.ibm.com.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corporation 2001, 2010.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book ix
Who should use this information ix
Related information xi

Summary of changes xiii

Part 1. The runtime environment 1

Chapter 1. Product overview 3
The runtime environment and BTS 3
The Service Flow Modeler 5
Adapter services 7
Server adapters 8
Distributed Programming Link (DPL) adapter 8
Link3270 server adapter 8
Front End Programming Interface (FEPI) server adapter 9
WebSphere MQ (WMQ) server adapter 9
Web service server adapter 10
Sequence of tasks for using CICS Service Flow Runtime and Service Flow Modeler 11
Benefits of CICS Service Flow Runtime and Service Flow Modeler 11
Service flow runtime terminology 12

Chapter 2. What's New 17

Chapter 3. Installing the CICS Service Flow Runtime 21
Software prerequisites 21
Performing post-installation tasks 22
Customizing the set up procedure DFHMAINJ 22
Running the product definition procedure DFHMASET 27
Copying the build time templates 28
Setting up data conversion 29
Configuring the autostart procedure for the Link3270 facility state cleanup programs 30
Adding support for BIDI transformation in service flows 31
CICS Service Flow Runtime samples listing 31
Running the installation verification procedure 33
Prerequisites for running the IVP 33
Starting the IVP 35
Responding to IVP errors 36
Using the Simulator program to verify the installation 36

Chapter 4. Planning 53
Discovery phase of an application transformation project 53
Planning phase of an application transformation project 53
Deployment patterns 53
Select the processing mode to use 55
How to run your business transaction request in asynchronous mode 56
Considering security 56
Using IBM WebSphere MQ-CICS bridge to control authentication 56
Setting the audit level 57
Determine if compensation is necessary 58
Deciding on whether to cancel or run compensating flow 60

© Copyright IBM Corp. 2001, 2010

Chapter 5. Deploying an adapter service . . . 63
Deploying a new adapter service . . . 63
Updating an existing adapter service . . . 64
The properties file . . . 65

Chapter 6. Invoking an adapter service . . . 67
The service requester . . . 67
Invoking an adapter service using a CICS-supplied interface . . . 68
Invoking an adapter service using WebSphere MQ triggering . . . 69
Invoking an adapter service from a Web service . . . 70
Sending the request message in containers . . . 71
Sending the request message in a COMMAREA . . . 73
Data conversion . . . 74
Data conversion using the WebSphere MQ interface . . . 74
Data conversion using a CICS-supplied interface . . . 75
Code page conversion . . . 76
Request message containers. . . . 76
Container DFHMAC-ALLPARMS . . . 76
Container DFHMAC-ERROR . . . 77
Container DFHMAC-LNK3270V1 . . . 78
Container DFHMAC-PASSTHRU . . . 78
Container DFHMAC-REQUESTV1 . . . 78
Container DFHMAC-SYSPARMV1 . . . 79
Container DFHMAC-USERDATA . . . 79
Container DFHWS-DATA . . . 79
Request message headers . . . 80
DFHMAH header structure . . . 80
DFHMAH field definitions . . . 81
DFHMAH2 header structure . . . 87
DFHMAH2 field definitions. . . . 88
CIA-SCREEN-HEADER header structure . . . 92
CIA-SCREEN-HEADER field definitions . . . 93
CIA-MAP-HEADER header structure . . . 94
CIA-MAP-HEADER field definitions . . . 95

Chapter 7. Managing adapter services . . . 99
Disabling access to adapter services . . . 99
Dumping the properties file . . . 99
Server runtime utilities . . . 100

Chapter 8. The runtime environment components . . . 103
Server runtime programs . . . 103
Server runtime files . . . 110
BTS data-containers . . . 114
Server runtime processing and the BTS NOCHECK option . . . 114
CICS Service Flow Runtime data-containers for aggregate processing patterns . . . 115
Data-containers for single connector processing patterns . . . 121
How CICS Service Flow Runtime programs use data-containers . . . 124

Chapter 9. Server runtime processing . . . 127
Synchronous and asynchronous processing . . . 127
Processing patterns. . . . 128
Single connector — persistent and nonpersistent . . . 128
Request processing patterns for Single connector — persistent and nonpersistent . . . 128

Reply processing patterns for Single connector — persistent and nonpersistent . . . 133
Aggregate connector — persistent and nonpersistent . . . 137
Request processing patterns for Aggregate connector — persistent and nonpersistent . . . 138
Reply processing patterns for Aggregate connector — persistent / nonpersistent . . . 143
Passthrough processing . . . 147
Processing patterns for a passthrough request . . . 148
Processing patterns for a passthrough reply . . . 153
Invocation of DPL in a FEPI or Link3270 service flow . . . 155
DPL server adapter processing . . . 156
FEPI server adapter processing . . . 157
CICS Service Flow Runtime FEPI file processing . . . 157
FEPI adapter LU assignment processing for non-unique UserIDs . . . 158
Link3270 server adapter processing . . . 163
Configuring the runtime environment to use transaction routing . . . 165
Facility state cleanup processing in the CICS Service Flow Runtime . . . 168
Managing shared temporary storage queues in a multiregion environment 170
Managing state cleanup for Link3270 server adapters . . . 171
Use restrictions for Link3270 bridge mechanism . . . 174
Web services server adapter processing . . . 176
WebSphere MQ server adapter processing . . . 177
Processing for non-unique user ids . . . 177
Why you should use unique user ids with CICS Service Flow Runtime 178
Managing state information . . . 179
Business state data management in aggregate connectors and single connector (persistent patterns) . . . 179
Business state data management in single connector (nonpersistent patterns) . . . 181
XML request and response processing . . . 182
XML request processing - non-passthrough . . . 184
XML response processing for non-passthrough . . . 186
XML request and response processing for passthrough . . . 188
How compensation processing works . . . 193
Error processing . . . 195
How errors are handled . . . 195
Error processing using TDQ . . . 196
Conditions that cause CICS Service Flow Runtime to write to the system console . . . 197

Chapter 10. Troubleshooting and support . . . 199
Learning more . . . 199
About troubleshooting . . . 199
About fixes and updates . . . 201
Troubleshooting checklist . . . 202
Troubleshooting post-installation errors . . . 203
Analyzing the properties file . . . 204
Troubleshooting Link3270 server adapters . . . 206
Troubleshooting the Web services server adapter . . . 208
Using a BTS audit trail for problem determination . . . 209
Debugging your application to facilitate problem determination . . . 209
Using CICS dump for problem determination . . . 210
Using CICS trace for problem determination . . . 210
Using CBAM for problem determination . . . 211
Troubleshooting aids . . . 211

The DFHMAERF error file 211
Vector logging 212
Trace points 213
Error messages 220
CICS Service Flow Runtime VSAM file error messages 220
Temporary storage queue (TSQ) error messages 224
Data-container error messages 226
DPL server adapter error messages 227
FEPI server adapter error messages 228
WebSphere MQ call error messages 231
Business Transaction Services (BTS) error messages 233
Link3270 server adapter error messages 236
Additional error messages 240
API error messages 244
CIA082*xx* update properties file error messages 248
XML parsing error messages 250
The IVP messages 253
Abend error messages 255
CIAI*xxxx* error messages 255
Abends 257
Applying APARs 258

Part 2. Samples 263

Chapter 11. Samples 265
JCL 265
Properties file update JCL (DFHMAMPU) 265
Properties file dump JCL (DFHMAMPD) 267
Properties file record descriptor (DFHMARPF) 268
Error file dump JCL (DFHMAMED) 269
Audit file dump JCL (DFHMABAP) 269
BTS Repository file dump JCL (DFHMABRP) 270
Link3270 Repository file update JCL (DFHMAMLU) 270
Link3270 Vector Log file dump JCL (DFHMAMVD) 271
Formatted file dumps 272
Properties file dump 272
Error file dump 283
Vector file dump 312
Conversion templates 316
DFHMADPL conversion template 317
DFHMADPP conversion template 317
IVP sample listing 318
The IVP back-end transactions that are provided 318
The IVP adapter programs that are provided 319
The IVP Simulator programs that are provided 320
The IVP sample JCL and procedures that are provided 321
Test data for IVP back-end transactions 321
Data added to the Customer Information file: DFHMABCF 321
Data added to the Customer Name (Phonetic) file: DFHMABNF 322
XML message formats 322
XML message formats for non-passthrough 322
XML message formats for passthrough requests 340

Part 3. Appendixes 363

Appendix. Migrating from MQSeries Integrator Agent for CICS Transaction Server (MQIAC) 365
Deploying MQIAC adapter services 365
Request processing of MQIAC adapter services 365
Link3270 bridge facility assignment processing for non-unique user ids. 366
Implementing facility assignment processing for non-unique UserIDs with Link3270 bridge server adapters 367
Run time processing of non-unique user IDs using Link3270 bridge server adapters 367
Error processing using WebSphere MQ 372
Conditions for writing to the system console. 373
Format of MQGET error written to console 374

Glossary 375

Index 389

Notices 393
Trademarks. 394

Sending your comments to IBM 395

About this book

This manual describes the CICS® Service Flow Runtime.

The CICS Service Flow Runtime User's Guide provides instructions on how to install, configure, and manage the runtime environment.

It contains the following parts:

- **Part 1 - The runtime environment**. This part contains installation, configuration and administration tasks, as well as descriptions of the runtime components and processing. The information is intended to provide the systems administrator, the developer and the business analyst with a sound understanding of the CICS Service Flow Runtime environment.

  This part contains information on the following items:

  – An introduction to CICS Service Flow Runtime V3.1, including essential terminology to help you understand key concepts. The relationship with the Service Flow Modeler is also described, which is the tool that developers use to create, generate and deploy the *Adapter services* that run with the runtime environment.

  – Installation and customization procedures that enable you to set up and configure the runtime environment. This includes an installation verification procedure (IVP).

  – Defining an Adapter service to CICS

  – A detailed description of the runtime components, the content and layout of the request message that is used by a service requestor to invoke CICS Service Flow Runtime processing, and the different types of processing that can take place.

  – A comprehensive list and description of CICS Service Flow Runtime programs, files and utilities, as well as descriptions of how the various types of Adapter services use BTS data-containers.

  – Problem determination and error diagnostics

  – A list of all system error messages and the recommended user responses.

- **Part 2 - Samples**. This part contains samples of the following:

  – JCL

  – Formatted file dumps, with field descriptions that can be used for diagnostics. CICS Service Flow Runtime includes formatted dumps of the following files:

    - Properties file dump
    - Error file dump
    - Vector file dump

  – Conversion template samples

  – Sample test data for IVP back-end transactions

  – XML message format samples

Who should use this information

The information in this book is intended for users who can be categorized into the following roles:

- **Decision making**, which can include the following types of roles:

© Copyright IBM Corp. 2001, 2010

  - **Executive** who champions an idea, product or plan of action and for making final business decisions with high impact on the resulting financial position of the company.
  - **Enterprise architect** who creates and maintains the overall technical (hardware and software) direction and technical infrastructure of the enterprise IT environment.
- **Development**, which can include the following types of roles:
  - **Business analyst** who identifies customer and business needs, captures requirements (often in the form of natural language, or free-form text), creates business use cases, and looks for areas of optimization – especially those that can be automated by software.
  - **Application developer** who develops the business services according to the Solution Architect's model, concentrating on the functional aspects of the application. This role may also be responsible for the creation of solution prototypes to validate ideas and business needs. The application developer will have a sound understanding of both the business goals and the programmatic functions of the existing applications from which he will create new services.
- **Operations and administration** which can include the following types of roles:
  - **Solution deployer** who is responsible for installation and deployment of newly developed or modified business solutions (releases) into existing IT infrastructure of an organization. It is likely that the solution deployer will bridge the gap between the deployment operations defined and run from the Service Flow Modeler tool and those deployment tasks related to defining the deployed Adapter service to the CICS Service Flow Runtime environment.
  - **Solutions architect** who is responsible for administering one or more running solutions and the corresponding solution-specific artifacts. This includes managing how well the solution is meeting its business goals and objectives (e.g. inventory turn was supposed to improve three fold, number of support calls from customers was supposed to be cut in half), or whether unforeseen business-oriented inefficiencies or issues have been introduced or evolved during the life of the current deployment (e.g. order processing bottlenecks induced by aggressive fraud detection mechanisms, fulfillment issues induced by the introduction of a new product line, call center order status updates experiencing a lag that causes problems in return processing). In terms of CICS Service Flow Runtime, the solution architect will need to manage and evaluate the effectiveness of newly developed services that are integrating with existing applications.
  - **Systems administrator** who is responsible for applying required maintenance to systems and software. It is likely that the systems administrator will perform the post-installation tasks, including customizing the runtime software to the environment and running the IVP.

    It is assumed that the system administrator will have knowledge in the following areas:

    - CICS
    - CICS business transaction services
    - Service Oriented Architecture (SOA)

It would be helpful if systems administrators are familiar with the IBM® WebSphere® MQ products.

Related information

There are several user assistance modules that you can use in conjunction with the CICS Service Flow RuntimeUser's Guide:

- View the Service Flow Modeler help to understand the how to create, generate and deploy Adapter services to the CICS Service Flow Runtime. It contains concept, task and reference information, as well as a sample. The information in the help describes how to create Adapter services and describes how organizations can implement Adapter services to expose their existing applications as a service-like interface, facilitating the move to service oriented architecture (SOA).

  The Service Flow Modeler is available through the free CICS Service Flow Feature.
- Read the *CICS Transaction Server for z/OS 3.1 Program Directory* for information concerning the material and procedures associated with the installation of CICS Service Flow Runtime modules.

Having access to the following WebSphere MQ resources will help you administer and manage CICS Service Flow Runtime:

- *WebSphere MQ System Management Guide*, for information on administering the WebSphere MQ-CICS bridge.
- *WebSphere MQ Application Programming Guide* and the *WebSphere MQ Application Programming Reference* for application programmers who want to code the service requestor to use the WebSphere MQ-CICS bridge.
- *WebSphere MQ for OS/390 System Management Guide*, for information on installing, configuring and implementing security for the WebSphere MQ product.

See the WebSphere MQ product family web site at http://www.ibm.com/software/ts/mqseries/.

Having access to the following CICS and CICS/BTS resources will help you customize, manage, administer, understand and troubleshoot the CICS Service Flow Runtime:

- *CICS Resource Definition Guide*, for defining CICS Service Flow Runtime CICS resources (programs, transactions and files) to your CICS system.
- *CICS Business Transaction Services* for information about CICS business transaction services (BTS) of CICS Transaction Server for OS/390®.
- *CICS Application Programming Reference*, for RESP and RESP2 values.
- *CICS Front End Programming Interface User's Guide*, for information on installing and configuring the Front End Programming Interface (FEPI) of CICS Transaction Server for z/OS and for information on writing FEPI application programs.
- *CICS Family: Client / Server Programming*, for application programmers who want to use the ECI to pass application data to the CICS Service Flow Runtime environment.
- *CICS External Interfaces Guide Version 2 Release 2* for application programmers who want to use the EXCI call interface to pass application data to the CICS Service Flow Runtime environment.

See the CICS product web site at http://www.ibm.com/software/htp/cics/. By following the links from this web site you can:

- Obtain the latest information about the CICS product.
- Access the most recent editions of CICS books in PDF format.

- Download CICS SupportPacs.

Summary of changes

This section briefly explains the changes that have been made for the following recent releases.

CICS Service Flow Runtime V3.1

Product renaming and availability

The product has been renamed to CICS Service Flow Runtime and is available as part of the free CICS Service Flow Feature.

The feature includes the Service Flow Modeler tooling and the enhanced CICS Service Flow Runtime code.

Changes to post-installation setup

The post-installation setup and installation verification procedures (IVP) have been simplified to enable you to quickly install the product and verify that the runtime environment is correctly configured. New samples have been created to run the set up and the IVP, as well as performing the necessary customization on the runtime libraries once created.

In addition, new messages and abend codes have been added for the IVP to help with identifying problems in the runtime environment.

New transaction

A new supplied transaction CMAA is provided to run the IVP.

Tracing

Tracing has been added to help with diagnosing problems.

Support for bidirectional languages

BIDI support has been added through the addition of two modules, FEJBDTRN and FEJBDTRE. The modules are called dynamically, enabling updates to the modules to occur without requiring you to regenerate or recompile the service flows. A new error message has also been added to help you diagnose BIDI problems.

APAR PK32131 has introduced a number of additional enhancements:

Support for WebSphere Developer for System z® version 7

With the release of WebSphere Developer for System z version 7, CICS Service Flow Runtime has been updated to support adapter services that are modeled and deployed in this new release of the tooling. The Service Flow Modeler is now available in the Enterprise Service Tools perspective of the tooling.

Support for outbound Web services in adapter services

You can now use the existing support in CICS to invoke a Web service from an adapter service, as well as exposing an adapter service as a Web service. To support outbound Web service requests, there is a new Web services server adapter called DFHMASWS. This runs under transaction CMAO and enables an adapter service to send Web service requests to service providers using the existing Web services support in CICS. The adapter sends Web service requests using a requester mode pipeline in the CICS region.

When you model the flow with an outbound Web service request, you specify which pipeline pickup directory to save the generated Web service binding file. Sample pipelines are provided to allow you to quickly deploy

© Copyright IBM Corp. 2001, 2010

adapter services that are making Web service requests or are acting as Web service providers. The required PIPELINE resources are created as part of the post-installation steps.

Outbound Web services are only available when you model, generate and deploy flows using WebSphere Developer for System z version 7.

Support for channels and containers
You can now invoke CICS Service Flow Runtime using a channel and containers when using a distributed program link (DPL). Several combinations of containers are allowed depending on your requirements. You can either use a simplified message header structure to invoke the deployed adapter service, or pass in the DFHMAH header structure and application data in one container, in the same way as a COMMAREA. Containers can also be used for passthrough processing.

A new DPL server adapter
The DPL server adapter DFHMASDP performs all distributed program links for an adapter service that has been generated using WebSphere Developer for System z version 7. It runs under transaction CMAS.

Improvements to the management of adapter services
A BTS PROCESSTYPE resource is now defined for every adapter service. The resource name matches the request name of the flow. This enables you to manage adapter services once they are deployed and running in CICS Service Flow Runtime, by installing, enabling, disabling, and discarding the PROCESSTYPE resource for a particular adapter service.

The job to update the properties file has a new **MODE** parameter. There are two values for this parameter:

OVERWRITE
This value specifies that you want to overwrite any existing adapter services or server adapters that might already be defined in the properties file with the same name. For example, this value would be useful if you are updating an existing adapter service.

SAFE This is the default value. When you run the job to update the properties file to deploy a new adapter service, and an adapter service already exists with the same name, you get an error and the adapter service is not defined in the properties file.

If you specify any other value, or if you specify no value at all, the default is assumed. This also applies to all adapter services that are already deployed in your CICS region.

Validation of the post-installation procedure DFHMAINJ
The parameter values that you specify in the post-installation job DFHMAINJ are now validated. This ensures that any problems are highlighted immediately, before the runtime libraries are customized. The job output also includes additional error messages to provide you with information on which parameter is in error. If there is an error in a parameter value, the libraries are not customized.

Improvements to vector logging
The vector logging function now uses two files, DFHMALVA and DFHMALVB, instead of one file (DFHMALVF) to record the flow of information through a Link3270 server adapter. One file is always active, and the other is either empty or contains old data. When DFHMALVA is full, the vector logging automatically swaps to use DFHMALVB. When

DFHMALVB is full, the vector logging swaps back to DFHMALVA, deleting the old content before beginning to write to the file. This means that the entire flow of data can now be analyzed for a Link3270 server adapter.

A new process for applying maintenance to the runtime environment
There is a new method for applying all maintenance to the runtime environment, using a supplied batch job called DFHMAINA.

Additional diagnostics
There are new messages and additional trace points to help with troubleshooting problems.

CICS Integrator Adapter for z/OS®

Relationship of CICS Integrator Adapter for z/OS to MQSeries® Integrator Agent for CICS Transaction Server version 1.1.3
Although most of the functionality that exists in MQSeries Integrator Agent for CICS Transaction Server version 1.1.3 exists in CICS Integrator Adapter for z/OS, CICS Integrator Adapter for z/OS is packaged with CICS Transaction Server version 3.1.

The CICS Integrator Adapter server runtime supports flows that were developed using MQSI Agent for CICS version 1.1.3.

Areas of information that apply to the MQSeries Integrator Agent for CICS Transaction Server version 1.1.3 only noted.

In previous editions of this guide, readers were sent to the *Adapter builder* help and to the *Using Control Center* documentation for information on the tool. The build time tool that supports CICS Service Flow Runtime V3.1 has migrated to the eclipse platform and exists as a component of the WebSphere Developer for System z product. The builder tool is named Service Flow Modeler and all the supporting information exists as a documentation plug-in in the WebSphere Developer for System z help.

New request message formats for passthrough processing
New request message formats for passthrough processing are supported. See “Request message headers” on page 80 for descriptions of the new message headers.

New deployment patterns
Additional deployment patterns are supported for adapter request processing. See “Deployment patterns” on page 53 for descriptions of the additional deployment patterns supported by the runtime.

Passthrough processing
Passthrough processing is supported. See “Deployment patterns” on page 53 for a description of how the runtime supports passthrough requests.

XML request and response messages
The runtime can handle XML request messages from a service requestor and can send XML response messages to the service requestor. For inbound requests, both the message header and the application data can contain XML formatted content.

See “XML request and response processing” on page 182 for a description of how the runtime handles request messages that are sent in XML format.

New programs
Several new programs have been added to the runtime. Several programs from MQSeries Integrator Agent for CICS Transaction Server have been modified to support new functionality.

See “Server runtime programs” on page 103 for a list of all the programs and for a description of how they function.

BTS data-container support

BTS data-container support for the additional deployment styles has been added.

See “BTS data-containers” on page 114 for information on the different implementations of data-containers used in the runtime environment.

The BTS NOCHECK option

The BTS `NOCHECK` option for request processing is supported.

See “BTS data-containers” on page 114 for information on using the BTS `NOCHECK` option.

Changes to error processing

Error processing is now configurable.

See “Error processing” on page 195 for information on the different types of error processing support in the runtime.

Changes to support for the Link3270 bridge mechanism

The use restrictions for the Link3270 bridge mechanism have been updated.

See “Use restrictions for Link3270 bridge mechanism” on page 174.

Changes to problem determination

The error messages and problem determination sections have been updated to support new functionality.

See “Error messages” on page 220 and Chapter 10, “Troubleshooting and support,” on page 199

Changes to the manual

- Complex tables have been modified for accessibility.
- The term *controlling application* has been changed to *service requester* throughout this guide.

Part 1. The runtime environment

This section describes how to install, configure and administer the runtime environment. It also describes the server runtime components and processing.

© Copyright IBM Corp. 2001, 2010

Chapter 1. Product overview

CICS Service Flow Runtime V3.1 (CICS SFR) is the server runtime environment to the adapter services that are modeled, generated and deployed using the Service Flow Modeler tool.

Service Flow Modeler is available in WebSphere Developer for zSeries® V6.0.1, WebSphere Developer for System z V7.0, and IBM Rational® Developer for System z V7.1. CICS Service Flow Runtime V3.1 supports adapter services that are generated by all these tools, although it is recommended that you use IBM Rational Developer for System z where possible.

An adapter service is a reusable composed business function that exposes a programmatic interface to a service requester in an Enterprise Information System (EIS). Adapter services are implemented in two stages:

- At build time, a developer uses the Service Flow Modeler for the following tasks:
  - Model a newly composed business service - or flow - using processes or services and their interfaces.
  - Capture existing EIS interfaces, such as screens or communication areas.
  - Generate adapters to and from interfaces, supporting both the accumulation of information that is used in request and response processing, as well as data transformation activities between the flow and the interface.
  - Expose business flows as a service.
- At run time, a service requester invokes the deployed adapter service to perform a business function as modeled in Service Flow Modeler. By composing EIS interfaces and exposing the resulting adapters as services, 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).

Depending on how the adapter service is modeled, it can contain a wide variety of functionality, such as control flow, data flow, sequential navigation, conditional branching including decision and iteration, data typing, storing data context, transformation of data elements, logical operations and custom code. CICS Service Flow Runtime is able to support this by using CICS Business Transaction Services (BTS).

The runtime environment and BTS

BTS consists of an application programming interface and support services that simplify the development of business transactions. It also allows you to control the execution of complex business transactions. Each action that makes up the business transaction is implemented as one or more CICS transactions.

All the processing associated with completing the business transaction are modelled at build time and then executed at run time. It is at run time that the deployed adapter service becomes operational. In BTS, an instance of a running business transaction is called a *process*. A process is a collection of one or more BTS activities. An activity is a basic unit of BTS execution and it is mapped to a traditional CICS transaction.

There is also a top-level program that is called the *root activity*, and this is used to control the overall progress of the business transaction. The root activity in CICS Service Flow Runtime is the Navigation Manager (DFHMAMGR). When the adapter

© Copyright IBM Corp. 2001, 2010

service is deployed to the CICS Service Flow Runtime, it exists as a process that is organized hierarchically. Data is exchanged using BTS *data-containers*. Data-containers are named areas of storage that are associated with a particular process or activity, and are maintained by BTS.

An adapter service is invoked when a service requester sends a request message to the CICS Service Flow Runtime. This is received by a stub program that initiates the Navigation Manager DFHMAMGR as the root activity. The Navigation Manager uses information contained in the request message to initiate the programmatic functions of the adapter service. These programmatic functions are known as *server adapters*, and are initiated using *request processing*. Request processing can vary, depending on the deployment pattern of the adapter service.

The following figure shows the components that are used when a service requester invokes an adapter service that is comprised of more than one server adapter.

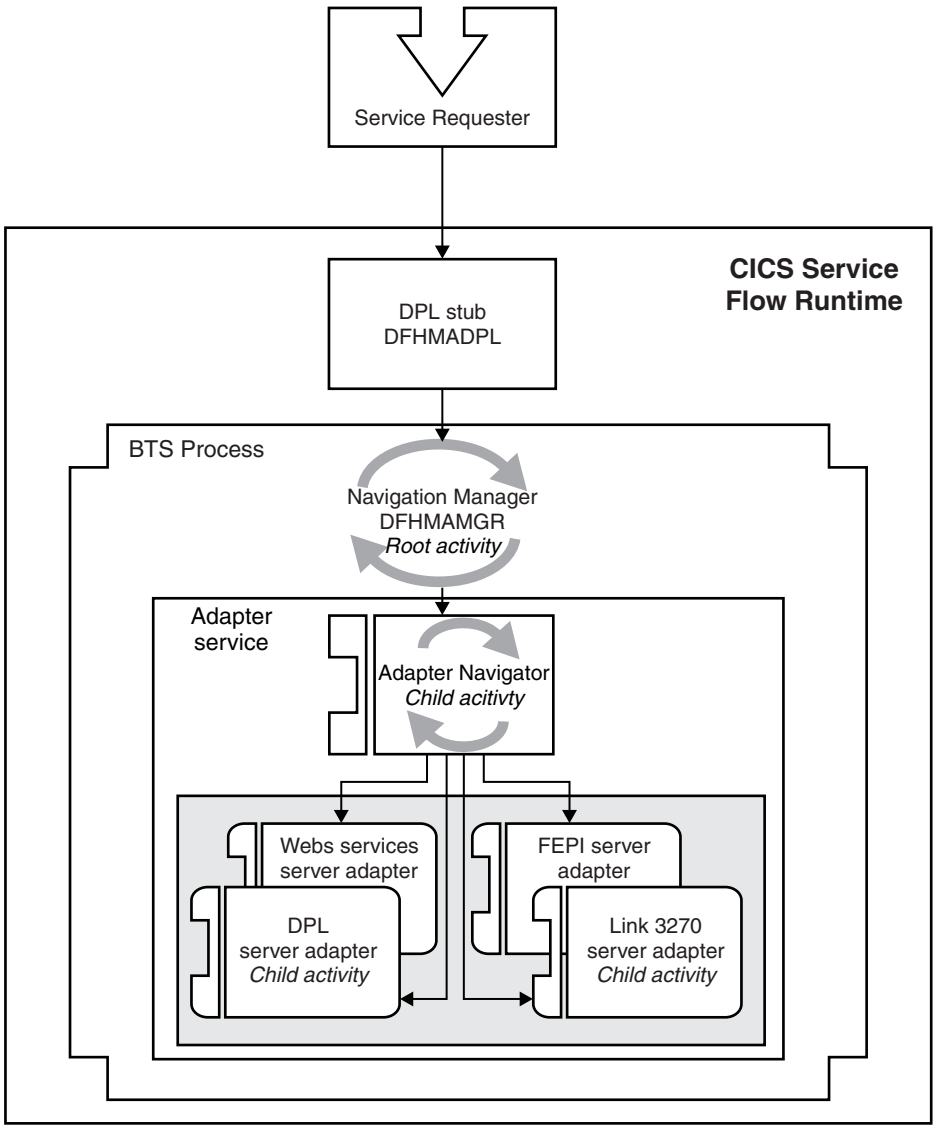


Figure 1. The components used in request processing for a complex adapter service

With CICS Service Flow Runtime, the functionality modeled using the Service Flow Modeler enables any application that is capable of initiating a CICS program to access:

- Existing CICS transactions using a Distributed Program Link (DPL).
- CICS and IMS applications using a 3270 data stream.
- WebSphere MQ-enabled applications using WebSphere MQ.
- Web services

Several examples of adapter services that implement business transactions are:

- Adding a sales order.
- Checking an account balance.
- Updating a customer record.

The Service Flow Modeler

Service Flow Modeler is a tool that enables you to create new services from existing applications in which the enterprise has substantial investments in time and money.

The tool enables you to create new services of existing applications in which the enterprise has substantial investments in time and money. You are recommended to use Service Flow Modeler in IBM Rational Developer for System z to use the latest set of enhancements in the runtime environment.

The Service Flow Modeler consists of several major components that are shown in the following figure.

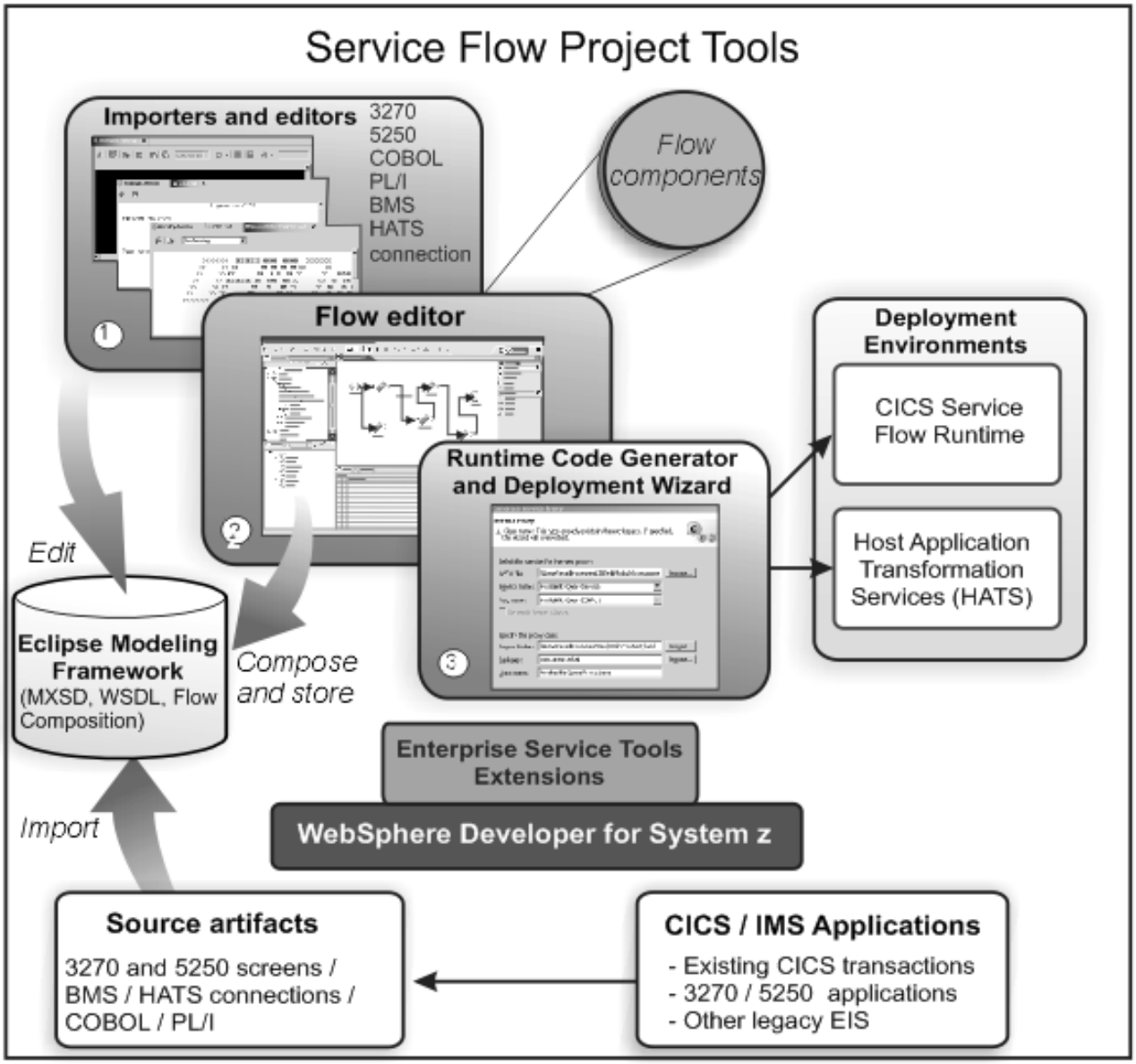


Figure 2. Components of Service Flow Modeler

Importers and editors

The importers enable you to input application resources from an existing Enterprise Information System (EIS) for the purpose of representing these resources within a common information model. The resources that can be imported include:

- 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 sequence flows and saving work.

The Flow editor

The Flow editor allows you to manually construct an adapter service that represents a dialogue, as well as populate a sequence flow using a

captured WSDL dialogue. Alternatively, you can use the Flow editor to annotate a flow with alternative paths of the host application that represent error paths, as well as additional business 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 an Adapter service that can be deployed into CICS Service Flow Runtime.

For more information about Service Flow Modeler, see the relevant help sections in the tooling product.

Adapter services

An adapter service is the set of files, programs and definitions that are generated by Service Flow Modeler and deployed in CICS Service Flow Runtime. It represents the service flow that was modeled, and contains the necessary programs to run that flow when it is invoked by a service requester.

An adapter service is comprised of one or more server adapters, that each represent a particular function that was modeled in the flow. For example, FEPI screen navigation would generate a FEPI server adapter. The adapter service adheres to a deployment pattern, that describes whether it is simple or complex.

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

Depending on the deployment pattern of the adapter service, the processing that takes place in the runtime environment varies, but also adheres to a processing pattern. If the adapter service has more than one server adapter, an Adapter Navigator is required to manage the order in which the server adapters are run and the results of their processing. If the adapter service is simple, then the Adapter Navigator is not required and the server adapter is run directly by the Navigation Manager.

You can also define that the adapter service should run asynchronously, if you're using WebSphere MQ, or synchronously for any supported interface. You can also set the persistence to log the processing in a BTS repository.

In addition to the supported types of server adapters, you can initiate custom programs using an **EXEC CICS LINK** command to augment the functionality of the adapter service. If you want to include behavior in your adapter service that you cannot directly model in Service Flow Modeler, 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 is used to invoke the custom program is the same as the mechanism used to initiate a CICS application using a distributed programming link.

Server adapters

A server adapter is a programmatic function of an adapter service that is invoked during request processing. Depending on the sequence of the modeled flow, the generated adapter service 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.

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

- Distributed Program Link (DPL) adapters
- Front End Programming Interface (FEPI) adapters
- Link3270 Bridge adapters
- WebSphere MQ (WMQ) adapters
- Web service adapters

Distributed Programming Link (DPL) adapter

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

If you are modeling a flow using WebSphere Developer for System z version 6, a DPL server adapter is generated for every distributed programming link. If the flow is simple, and only generates a single DPL server adapter, it is handled directly by the navigation manager at run time. If the flow is complex, and generates many server adapters, these are controlled using an adapter navigator.

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.

If you are modeling a flow using WebSphere Developer for System z version 7, a standard DPL server adapter called DFHMASDP is invoked for all distributed programming links in the adapter service. As DFHMASDP is handling all of the distributed programming links, it is always invoked by an adapter navigator and runs under the CMAS transaction. The processing that takes place when DFHMASDP is invoked is the same as existing DPL server adapters generated by WebSphere Developer for System z version 6 flows.

Link3270 server adapter

The Link3270 server adapter enables a service requester to conduct an interactive BMS request and reply dialog with 3270 application programs that are running in CICS by using the CICS Link3270 bridge mechanism.

3270 emulation and navigation logic for 3270 application screens are contained within the Service Flow Modeler, and are generated as a Link3270 server adapter. This server adapter can comprise the complete adapter service generated by a flow that conforms with the single connector simple pattern, or it could be one of many server adapters that comprise the adapter service generated by a flow that conforms with the aggregate connector complex pattern.

The navigation logic is composed of the following types:

- **Static navigation** consists of logic to traverse a fixed series of screens, where the screen input and function key (i.e. Enter, PF1, etc.) are known ahead of time.

- **Variable navigation** is required when data returned from a response screen is used to determine field input or function key for the next request.
- **Response formatting** moves and formats the business data from the response screens to the business client response area (output data-container).

Typical 3270 application programs use CICS Basic Mapping Support (BMS) commands, such as **SEND MAP** and **RECEIVE MAP**, to communicate with the terminal user. The BMS commands reference a BMS symbolic map, that contains field data. For a non-3270 terminal client, the Link3270 server adapter provides a virtual terminal environment for the 3270 application, by intercepting BMS and terminal commands issued by the application.

In a web environment, the application program communicates with a Java™ client instead of a 3270 terminal. The application program remains unchanged and continues to use a BMS symbolic map as an interface to the Java client.

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. However, instead of using a screen of data, the server adapter views the BMS symbolic map, or application data structure (ADS), used within the target 3270 application program. The Link3270 server adapter also indicates which attention identifier (enter or function key) was pressed, and can optionally indicate which field the cursor was in when the key was pressed.

Front End Programming Interface (FEPI) server adapter

Service Flow Modeler has a tool that conducts an interactive 3270 request and reply dialog with CICS and IMS applications. Using FEPI, it sends requests to and receives replies from any CICS or IMS application whose 3270 datastream is intended for a SLU2 3278 Model 2 terminal (24 rows by 80 columns).

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

- Begin the FEPI session
- Parse screens sent by the CICS or IMS application
- Identify the screen and its fields, attributes and data
- Construct and sends an appropriate reply, based on the modeling and on simple business logic
- Handle the next screen by parsing, identifying and constructing a reply or keystroke
- Manage state information about the status of LUs
- End the FEPI session.

The buffer in a single send or receive must not be greater than 3,600 bytes.

WebSphere MQ (WMQ) server adapter

The WMQ server adapter interfaces with WebSphere MQ-enabled applications. Any adapter service that includes a WMQ server adapter adheres to a complex deployment pattern. So server adapters of this type are always run by the Adapter Navigator.

When an adapter service is generated from a flow that includes a WMQ node, two server adapters are generated.

- The Websphere MQ PUT server adapter is responsible for putting messages on the target queue.
- The WebSphere MQ GET server adapter is responsible for retrieving a response message from the target queue.

The definition of the server adapters in the properties file define what kind of interaction is required during request processing. The MP-MQ-MSGTYPE parameter for the server adapter record defines what action the Adapter Navigator should take. Depending on what you model for the WMQ node in the flow, the WMQ GET server adapter program does not have to run. For example, if you model that the application data should be sent to a target application and no response is required, then the GET server adapter does not run. In this case the Adapter Navigator moves to the next server adapter in the request processing, as modeled in the flow.

Web service server adapter

The Web service server adapter performs outbound Web service requests using the existing Web services support in CICS. Web service adapters are only available when you model, generate and deploy flows using WebSphere Developer for System z version 7.

The Web service server adapter is called DFHMASWS and runs under the CMAO transaction. DFHMASWS handles all outbound Web service requests, so it is always invoked by the Adapter Navigator. DFHMASWS sends a 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. DFHMASWS waits for a response from the Web service provider and then returns to the Adapter Navigator.

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

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 HFS file which contains an XML description of the message handlers and their configuration. This 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 should use, as well as 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 also provided, and this is the default pipeline that is used by Service Flow Modeler. 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.

Note: Do not configure this pipeline to include additional handlers. DFHMASFR can be updated when APARs are applied or the product is reinstalled, and you

would lose your configuration changes.

If you want to perform additional processing on Web service requests in the pipeline, you can either:

- 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 Modeler

The following sequence lists the tasks that you would typically perform when using Service Flow Modeler and CICS Service Flow Runtime.

1. At build time:
   a. Plan and design your implementation.
   b. Understand, design and implement the service requester.
   c. Create the adapter service using the Service Flow Modeler, this could include the following tasks:
      1) capturing existing EIS interfaces.
      2) modeling a newly composed business service using these interfaces.
      3) generating a new adapter service.
      4) deploying the adapter service to the supported runtime environment.

   For detailed information on the task flow for using the Service Flow Modeler, see the applicable sections of the help in the WebSphere Developer for System z tooling.
2. Preparing 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 on to your z/OS server.
   c. Set security and authentication parameters.
   d. Perform the necessary post installation procedures.
   e. Optional: Run the installation verification procedure (IVP).
3. At run time:
   a. Deploy adapter services in the runtime development region on the z/OS server.

      Note: This assumes that if required, FEPI, WebSphere MQ, MRO, ISC, IRC and CICS transactions and custom programs, associated software and systems have already been made operational and available.
   b. Monitor performance of the CICS Service Flow Runtime.
   c. Address any problems.

Benefits of CICS Service Flow Runtime and Service Flow Modeler

CICS Service Flow Runtime and Service Flow Modeler provide the following benefits:

- *Intelligent and efficient reuse of critical IT assets :* Service Flow Modeler and the CICS Service Flow Runtime enable an organization to unlock critical IT assets (applications and data) from their current legacy status, and re-purposes those

assets to participate in a service oriented architecture (SOA). This process is sometimes referred to as *application transformation*.

When an organization needs to create a new process for doing business with partners, suppliers, customers, or employees, Service Flow Modeler and CICS Service Flow Runtime provide tools and supported runtime modules that allow the organization to leverage the value they already have in their existing enterprise information systems and to use these systems for service oriented business processes. This process of transforming existing applications is more efficient than creating new applications.

- *Comprehensive toolset enabling both analysis of existing applications and visual representation of business functions:* The Service Flow Modeler is comprised of a toolset that allows developers to analyze existing green screen and application logic flow for understanding where the capabilities lie within their existing assets and provides the foundation for plans on how to leverage the applications in new processes, services, or offerings.
- Service Flow Modeler makes use of the Eclipse Integrated Development Environment (IDE).
- The Service Flow Modeler generates all modeled code, runtime properties and compilation JCL.
- The Service Flow Modeler and the CICS Service Flow Runtime provide an efficient sequence flow integration within the z/OS environment.
- The CICS Service Flow Runtime allows the navigation or sequence flow to be placed closer to the CICS and IMS™ EIS target applications, enabling multiple transactions or applications to be accessed with one request from the service requestor.
- The CICS Service Flow Runtime enables more efficient use of computing resources. It off loads work from the service requestor. At run time, instead of a service requestor invoking each transaction individually, it can invoke request processing that can:
  - Invoke the CICS and IMS transactions, CICS applications or WebSphere MQ-enabled applications
  - 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.
- Adapter services can be deployed to the CICS Service Flow Runtime without changing applications or business processes at all. Typically, all the integration work is performed in the CICS Service Flow Runtime.

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.

adapter service
: A reusable composed business function that exposes a programmatic interface to a service requester in an Enterprise Information System. An adapter service is the generated output from Service Flow Modeler.

basic mapping support (BMS)
: An application programming interface between CICS programs and terminal

devices. A BMS map set is made up of maps that specify how field data is to be formatted. BMS removes device dependencies from the application program. It interprets device-independent output commands and generates device-dependent data streams for specific terminals. It also transforms incoming device-dependent data into device-independent format. These features eliminate the need to learn complex device data streams. They also allow you to use the same program for a variety of devices, because BMS determines the device information from the terminal definition, not from the application program

build time
The time period when the service interface is defined, modeled or modified.

business transaction
A business entity that has been defined through business process analysis and that can be implemented using information technology. Typically, a business transaction maps to multiple CICS transactions. Developer's use the Service Flow Modeler to model business transactions as *adapter services*, which they then can deploy to the runtime environment.

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. How this is implemented is decided by the application designer, but often means the undoing, or reversing, of the actions that the activity took.

compensation sequence flow
A directed graph that models the processing required should a failure occur at run time. Although Service Flow Modeler does not support creating compensation service flows explicitly, a programmer can set values in a service flow and use logic in the controlling application to associate one flow to another for the purpose of performing compensation, as long as the controlling application provides the necessary information in the message header (DFHMAH).

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 as long as they all have different names.

deploy
To place files or install software into an operational environment. In J2EE, this involves creating a deployment descriptor suitable to the type of application that is being deployed. In the case of Service Flow Modeler, the deployment descriptor would define the components and operating system parameters of the adapter service.

deployment pattern
A well defined usage pattern that describes how a service should run in the target environment. Adapter services 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
Represents the data source in an enterprise information system, for example 5250 and 3270 screens, COBOL record descriptions and transactions. Using Service Flow Modeler, developers are able to model and compose these interfaces in to 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.

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 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.

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 that CICS business transaction services can work with, and 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 (it has no parent activity). The root activity normally 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.

run time
The time period during which the adapter service is available for invocation by a service requester.

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

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 within 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.

sequence flow
A graphical representation of a composed service. It shows a sequence of operations, assignments and conditionals that are linked into finite paths such that a request message is processed resulting in a response message.

Service Flow Modeler
An eclipse-based application integration tool set that enables developers to capture, model and expose adapters as services. It enables an organization to expose existing applications as a service-like interface, facilitating the move to a service oriented architecture (SOA).

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 transaction is the controlled interaction between two entities, usually involving the passing of information. Transactions enforce ACID properties (atomicity, consistency, isolation, and durability) in the runtime environment. In certain cases transactions can be rolled back, or reversed to a certain point.

transform
The process of changing the structure and values of data from one form to another. At build time, a developer can use Service Flow Modeler to transform existing interfaces in an EIS in order to facilitate participation of EIS applications in a service in an SOA.

Chapter 2. What's New

The functional enhancements and changes for this release are summarized below.

Product renaming and availability
The product has been renamed to CICS Service Flow Runtime and is available as part of the free CICS Service Flow Feature.

The feature includes the Service Flow Modeler tooling and the enhanced CICS Service Flow Runtime code.

Changes to post-installation setup
The post-installation setup and installation verification procedures (IVP) have been simplified to enable you to quickly install the product and verify that the runtime environment is correctly configured. New samples have been created to run the set up and the IVP, as well as performing the necessary customization on the runtime libraries once created.

In addition, new messages and abend codes have been added for the IVP to help with identifying problems in the runtime environment.

New transaction
A new supplied transaction CMAA is provided to run the IVP.

Tracing
Tracing has been added to help with diagnosing problems.

Support for bidirectional languages
BIDI support has been added through the addition of two modules, FEJBDTRN and FEJBDTRE. The modules are called dynamically, enabling updates to the modules to occur without requiring you to regenerate or recompile the service flows. A new error message has also been added to help you diagnose BIDI problems.

APAR PK32131 has introduced a number of additional enhancements:

Support for WebSphere Developer for System z version 7
With the release of WebSphere Developer for System z version 7, CICS Service Flow Runtime has been updated to support adapter services that are modeled and deployed in this new release of the tooling. The Service Flow Modeler is now available in the Enterprise Service Tools perspective of the tooling.

Support for outbound Web services in adapter services
You can now use the existing support in CICS to invoke a Web service from an adapter service, as well as exposing an adapter service as a Web service. To support outbound Web service requests, there is a new Web services server adapter called DFHMASWS. This runs under transaction CMAO and enables an adapter service to send Web service requests to service providers using the existing Web services support in CICS. The adapter sends Web service requests using a requester mode pipeline in the CICS region.

When you model the flow with an outbound Web service request, you specify which pipeline pickup directory to save the generated Web service binding file. Sample pipelines are provided to allow you to quickly deploy adapter services that are making Web service requests or are acting as Web service providers. The required PIPELINE resources are created as part of the post-installation steps.

© Copyright IBM Corp. 2001, 2010

Outbound Web services are only available when you model, generate and deploy flows using WebSphere Developer for System z version 7.

Support for channels and containers

You can now invoke CICS Service Flow Runtime using a channel and containers when using a distributed program link (DPL). Several combinations of containers are allowed depending on your requirements. You can either use a simplified message header structure to invoke the deployed adapter service, or pass in the DFHMAH header structure and application data in one container, in the same way as a COMMAREA. Containers can also be used for passthrough processing.

A new DPL server adapter

The DPL server adapter DFHMASDP performs all distributed program links for an adapter service that has been generated using WebSphere Developer for System z version 7. It runs under transaction CMAS.

Improvements to the management of adapter services

A BTS PROCESSTYPE resource is now defined for every adapter service. The resource name matches the request name of the flow. This enables you to manage adapter services once they are deployed and running in CICS Service Flow Runtime, by installing, enabling, disabling, and discarding the PROCESSTYPE resource for a particular adapter service.

The job to update the properties file has a new **MODE** parameter. There are two values for this parameter:

OVERWRITE
: This value specifies that you want to overwrite any existing adapter services or server adapters that might already be defined in the properties file with the same name. For example, this value would be useful if you are updating an existing adapter service.

SAFE This is the default value. When you run the job to update the properties file to deploy a new adapter service, and an adapter service already exists with the same name, you get an error and the adapter service is not defined in the properties file.

If you specify any other value, or if you specify no value at all, the default is assumed. This also applies to all adapter services that are already deployed in your CICS region.

Validation of the post-installation procedure DFHMAINJ

The parameter values that you specify in the post-installation job DFHMAINJ are now validated. This ensures that any problems are highlighted immediately, before the runtime libraries are customized. The job output also includes additional error messages to provide you with information on which parameter is in error. If there is an error in a parameter value, the libraries are not customized.

Improvements to vector logging

The vector logging function now uses two files, DFHMALVA and DFHMALVB, instead of one file (DFHMALVF) to record the flow of information through a Link3270 server adapter. One file is always active, and the other is either empty or contains old data. When DFHMALVA is full, the vector logging automatically swaps to use DFHMALVB. When DFHMALVB is full, the vector logging swaps back to DFHMALVA, deleting the old content before beginning to write to the file. This means that the entire flow of data can now be analyzed for a Link3270 server adapter.

A new process for applying maintenance to the runtime environment

There is a new method for applying all maintenance to the runtime environment, using a supplied batch job called DFHMAINA.

Additional diagnostics

There are new messages and additional trace points to help with troubleshooting problems.

Chapter 3. Installing the CICS Service Flow Runtime

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

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, referring to the *CICS Service Flow Runtime Program Directory*. The Program Directory also includes details on space and storage requirements.
3. Perform the post-installation steps. You must perform these steps for every CICS region that hasCICS Service Flow Runtime installed.
4. Optional: Run the installation verification procedure (IVP).
5. Restart the CICS region.

These tasks are explained in detail in the following sections.

See “Migrating from MQSeries Integrator Agent for CICS Transaction Server (MQIAC),” on page 365 if you want to run existing MQIAC flows in the CICS Service Flow Runtime environment.

The Service Flow Modeler that is used to create the Adapter services is part of the WebSphere Developer for System 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 for z/OS Version 3.1.

  CICS Business Transaction Services (BTS) must be configured 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 need to 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 170.
- IBM Enterprise COBOL for z/OS and OS/390 Version 3 Release 1 or later
- z/OS V1R4.0 Language Environment® or later
- **Optionally:** 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
  - WebSphere MQ Server Adapters
  - Simulator sample

If you want to use FEPI PassTickets:

- OS/390 Security Server (RACF®) Version 2.5, including PTFs UW91119 and UW91120, and PTF UW90545 for APAR OW35612.

  Note: You can use another type of external security manager as long it supports PassTickets.

© Copyright IBM Corp. 2001, 2010

Performing post-installation tasks

Post-installation tasks are what you do to customize and test the CICS Service Flow Runtime software that has been installed on to your z/OS server.

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

This section contains the following information:

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

The tasks are sequential and should be performed in the order in which they appear.

Customizing the set up 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 to customize the JCL, copy member DFHMAINJ from the .SCIZSAMP library to a new location. This ensures that any modifications to DFHMAINJ are not overwritten when system maintenance or version upgrades are applied.

Edit DFHMAINJ as follows:

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. For any parameters that you do not require, replace the example parameter values with blanks. Do not remove or comment out any of the optional parameters from the job. The list of parameters are described in “DFHMAINJ parameters” on page 24.
3. Optional: When APAR PK32131 is applied, 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.
4. Submit DFHMAINJ and check the output. You should see the following messages in the //SYSTSPRT of the DFHMAINJ job output:

```
CIAI1002I SCIZSAMP customization beginning.
CIAI1000I Validation of input parameters is taking place.
.
.
CIAI1011I SCIZSAMP customization ended without errors.
```

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

5. Optional: If you have a BTS repository already installed and defined in the CICS region, you can use this 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.

Three runtime libraries are created and the members copied into them. These libraries are

- .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 there is a problem with one of the parameter values in the job, no customization takes place. The job output contains one or more error messages explaining why the customization could not take place. After you have corrected the cause of the error, rerun DFHMAINJ to perform the customization on your runtime libraries.

The customized DFHMAINJ JCL parameters could look as follows:

```
*****************************************************************
JOB1        //++++++++ JOB ,CLASS=M,REGION=0M,
JOB2        //         NOTIFY=&SYSUID,MSGCLASS=H
JOB3        //*
*
SHLQ        ANTZ.DFHMA000.INC10
QUAL        WARDABL.ANTZTEST
VOLSER      P2P210
RDOLIST     CICSSFRL
MQ          YES
CSDNAME     WARDABL.ZED3.DFHCSD
HLQCICS     CTS310.CICS640
HLQCOBOL    PP.COBOL390.V330
HLQCEE      CEE
HLAPPLID    IYK2ZIO4
WSDIR_REQ   /hfs/wsbind/file/directory/structure/
CONFIG_REQ  /usr/lpp/cicsts/samples/pipelines/\
            \basicsoap11requester.xml
SHELF_REQ   /var/cicsts/
WSDIR_PROV  /hfs/wsbind/file/directory/structure/
CONFIG_PROV /usr/lpp/cicsts/samples/pipelines/\
            \basicsoap11provider.xml
SHELF_PROV  /var/cicsts/
*
*****************************************************************
*                                                               *
* Optional values.                                              *
*                                                               *
* NOTE:                                                         *
* If you specified "MQ YES" then you MUST specify a value for   *
* HLQMQ.                                                        *
```

```
*                                                               *
*****************************************************************
PREFIX      TEST
HLQMQ       MQM.V600
THE_QMGR    QG2C
CICS.INITQ  IYK2ZIO4.INITQ
HLSYSID     NIO4
*
/*
//
```

DFHMAINJ parameters

DFHMAINJ has mandatory and optional parameters that can be validated 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, as it 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 should be used 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 a-z 0-9 ./_ ($ # @

RDOLIST *grplist*
The name of the CICS RDO list that contains the CICS SFR group, and the MQ groups if required. It is recommended that you use the list CICSSFRL.

Acceptable characters:
A-Z a-z 0-9 $ # @

MQ YES|NO
Indicates if you require support for WebSphere MQSeries (WMQ). If you specify

MQ YES, then you must also define values for the **HLQMQ** and **THE_QMGR** parameters. You must specify this parameter if you want to use the Simulator program to verify your installation.

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.

HLAPPLID *ivp_cics_applid*
The APPLID of the CICS region that is used for running CICS SFR.

This value is also used to configure the pool and target name parameter properties for the FEPI adapter IVP.

Acceptable characters:
A-Z a-z 0-9 $ # @

Note that the first character of this value cannot be a number.

WSDIR_REQ */your/wsdir/requester/*
The fully qualified name of the Web service pickup directory on HFS 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 should not exceed 255 characters, and should 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 HFS. 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 should not exceed 255 characters, and should 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 HFS that contains subdirectories for the requester mode pipeline configuration files and Web service requester binding files. The length of the fully qualified directory name should not exceed 255 characters, and should 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 HFS 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 should not exceed 255 characters, and should 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 HFS. 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 should not exceed 255 characters, and should 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 HFS that contains subdirectories for the provider mode pipeline configuration files and Web service provider binding files. The length of the fully qualified name should not exceed 255 characters, and should 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, this would rename every jobname in the members of the runtime SCIZSAMP library with CSFR as the first four characters e.g. //DFHMASET would become //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.

HLQMQ *your.mq.hlq*
A 1-35 character length value that is the high level qualifier of the WMQ runtime libraries.

Note: This parameter is required if you specified `MQ YES`.

THE_QMGR *qmgrname*
A 1-4 character length value that is the name of the WMQ queue manager that resources are defined to.

Note: This parameter is required if you specified `MQ YES`.

Acceptable characters:
`A-Z a-z 0-9`

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

CICS.INITQ *your.cics.initq*
A 1-44 character length value that is the name of the CICS WMQ initiation queue.

Note: This parameter is required if you specified `MQ YES`.

Acceptable characters:
`A-Z a-z 0-9 ./_%`

HLSYSID *ivp_cics_sysid*
The SYSID of the CICS region that is used for the IVP. This is required if you want to run the IVP for the FEPI adapter.

Acceptable characters:
`A-Z 0-9 $ # @`

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, as well as creating the required CICS and WMQ resources.

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

1. Optional: If you intend to run the IVP for the FEPI adapter, you might need to change the FEPI pool and target name parameters. The value of the **HLAPPLID** parameter in DFHMAINJ is used to configure the pool and target name. If you have configured FEPI to use different values in your CICS region, edit the TYPE=3 entries in member DFHMASPS of the .SCIZSAMP library to reflect your pool and target name.
2. Run DFHMASET. The job executes these steps:
   - Compiles the file initialization program and the record delete program
   - Creates the following files:

DFHMAERF	Error file
DFHMAMPF	Properties file
DFHMALRF	Link Bridge repository file
DFHMALVA	Link Bridge vector log file
DFHMALVB	Link Bridge vector log file
DFHMAL2F	Link Bridge state file

DFHMACOF	FEPI SLU connection file
DFHMATIF	FEPI target interaction file
BTS	BTS file
DFHMALSF	Link3270 Bridge state file for MQIAC

- Initializes DFHMACOF and creates an alternate index DFHMAC1F for it.
- **Optional:** Executes DFHMASMQ to create the MQ resources. This only occurs if you specified `MQ YES` in the DFHMAINJ job.
- Updates the CSD to create the CICS resources.

3. Check the job output. All steps should have a return code of 0, except for the compiling of DFHMADCD, DFHMADCI, DFHMADUP, DFHMAEUP 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.
4. Optional: If you specified `MQ YES`, then check the output from the MQDEFS step. CSQUTIL can have a return code of 0, even if errors occurred.
5. Update CICS JCL to specify the runtime library SCIZLOAD in the DFHRPL concatenation.
6. Include the name of the CICS RDO list that contains the CICS SFR group in the **GRPLIST** system initialization parameter. This is the name that you used for the **RDOLIST** parameter in DFHMAINJ.

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.

It is recommended that you run the IVP to check that your setup is configured correctly and that all the necessary resources are available for each of the supported adapters. If you are not running the IVP, restart CICS to complete the set up of CICS SFR.

Copying the build time templates

Build time templates are customized during the setup of CICS SFR, and are required to deploy Adapter services in your development environment.

The build time templates are listed in Table 1. These templates are customized when you run the set up procedure DFHMAINJ. The customized versions should be copied to every client machine intending to generate and deploy Adapter services in CICS SFR.

You need to copy the build time templates from the runtime library SCIZSAMP to the following directory location on the client machine: *install_path*`/wdz/eclipse/plugins/com.ibm.etools.sfm.runtime.cia.common_`*x.x.x*`/templates`, where *install_path* is the directory path where you have installed WebSphere Developer for System z. The version number of the plug-in, indicated by *x.x.x*, depends on which version of the tool you have installed.

Table 1. CICS Service Flow Runtime build time templates

Name	Description
DFHMAXCJ	JCL to compile deployed server adapters

Table 1. CICS Service Flow Runtime build time templates (continued)

Name	Description
DFHMAXCP	Procedure to compile CICS programs. This procedure can also be used in IVP program compilations
DFHMAXPU	JCL to update the properties file
DFHMAXRD	JCL to define the generated Adapter service 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, it might be necessary to perform data conversion in the runtime routing region, utilizing a customized version of the standard CICS conversion table DFHCNV.

The customized CICS conversion table (DFHCNV), must specify an entry for the initial CICS Service Flow Runtime program name. This program name will be one of the following:

- DFHMADPP - The DPL program for passthrough processing.
- DFHMADPL - The DPL program for non-passthrough processing.

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

- 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.

  Module DFHMAXCV contains the DFHMADPP and DFHMADPL templates.

  See “DFHMADPP conversion template” on page 317 for a sample of the conversion template for passthrough processing and “DFHMADPL conversion template” on page 317 for a sample of the conversion template for non-passthrough processing. The conversion templates provide the CICS Service Flow Runtime message header structures and layout off-sets 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 provided in the samples library *cizhlq*.SCIZSAMP.

  If you choose this option, you must add this load library to the CICS RPL 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, found 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.

DFHMALSC performs cleanup functions on temporary storage queues (TSQs) for Link3270 adapter services of the single connector nonpersistent type.

DFHMALFC performs cleanup functions on the VSAM file DFHMAL2F for Link3270 adapter services of the following types:

- Persistent and nonpersistent aggregate connectors
- Persistent single connectors

You can start the cleanup programs using the program list table (PLT). In the final stages of CICS initialization, a set of programs can be executed as specified in the PLT. The programs shut down with CICS.

1. Define and assemble 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 the *CICS Customization Guide*. For information on defining a PLT, see the *CICS Resource Definition Guide*.

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 cause a negative performance impact. To prevent this, 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 the *CICS System Definition Guide*.

For a description of facility state cleanup processing on TSQs see “Facility state cleanup processing — TSQ” on page 169. For a description of facility state cleanup processing on the VSAM file, see “Facility state cleanup processing — VSAM” on page 169

Adding support for BIDI transformation in service flows

Service flows can optionally be configured to support BIDI transformation in the Service Flow Modeler. If these flows are required to run in CICS SFR, you must enable the runtime environment to support them.

There are two BIDI modules, FEJBTRN and FEJBTRE. The modules provide the same BIDI transformation functionality, but only FEJBTRN provides Arabic locale support. Also FEJBTRN is a .dll file, whereas FEJBTRE is a standalone module. These two modules are provided by WebSphere Developer for System z. Instructions on how to move the modules to your CICS region are included in the help of WebSphere Developer for System z.

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

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

In the event of a failed BIDI transformation, a CIA08008E error is recorded in the error dump file. Use the JCL DFHMAMED in the .SCIZSAMP library to run the utility DFHMAEUP, which formats the error dump file. The error dump has two parts - static content and dynamic content. Look in the dynamic part of the dump, which details specific fields and values that you can use to determine what kind of error has occurred.

CICS Service Flow Runtime samples listing

The samples that are provided in the .SCIZSAMP library are described in the following table. These samples are copied by the DFHMAINJ job to the runtime libraries that are used by CICS SFR. For a list of the samples used in the IVP, see “IVP sample listing” on page 318.

Table 2. CICS Service Flow Runtime samples

Name	Description
DFHMABAP	BTS Audit Log dump JCL
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

Table 2. CICS Service Flow Runtime samples (continued)

Name	Description
DFHMADDC	JCL to delete and define SLU Connection VSAM file
DFHMADDE	JCL to delete and define Error Log VSAM file
DFHMADDL	JCL to delete and define Link3270 State VSAM file
DFHMADDP	JCL to delete and define Properties VSAM file
DFHMADDR	JCL to delete and define Link3270 Repository VSAM file
DFHMADDT	JCL to delete and define Target Interaction VSAM file
DFHMADDV	JCL to delete and define Link3270 Vector Log VSAM file
DFHMADDX	JCL to delete and define all CICS Service Flow Runtime VSAM files
DFHMADEA	New Link3270 Vector Log file IDCAMS delete
DFHMADEC	SLU Connection file IDCAMS delete
DFHMADEE	Error file IDCAMS delete
DFHMADEL	Link3270 State file IDCAMS delete
DFHMADEP	Properties file IDCAMS delete
DFHMADER	Link3270 Repository file IDCAMS delete
DFHMADET	Target Interaction file IDCAMS delete
DFHMADEV	Old Link3270 Vector Log file IDCAMS delete
DFHMADFA	New Link3270 Vector Log file IDCAMS define
DFHMADFC	SLU Connection file IDCAMS define
DFHMADFE	Error file IDCAMS define
DFHMADFL	Link3270 State file IDCAMS define
DFHMADFP	Properties file IDCAMS define
DFHMADFR	Link3270 Repository file IDCAMS define
DFHMADFT	Target Interaction file IDCAMS define
DFHMADFV	Old Link3270 Vector Log 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.
DFHMAIV	The IVP mapset that is used by the IVP program DFHMAIVP
DFHMAIVP	The IVP program
DFHMAMED	Error file dump JCL
DFHMAMLU	Link3270 Repository file update JCL
DFHMAMMQ	CICS Service Flow Runtime WebSphere MQ resource definitions
DFHMAMPD	Properties file dump JCL
DFHMAMPU	Properties file update JCL
DFHMAMRD	CICS Service Flow Runtime resource definitions
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

Table 2. CICS Service Flow Runtime samples (continued)

Name	Description
DFHMASFR	Sample requester mode pipeline that can be used for Web service requests
DFHMASID	Data set that contains the WMQ resource definitions for CICS Service Flow Runtime
DFHMASIM	Creates the IVP resources
DFHMASMC	JCL program that specifies the instream `MQ` parameter for the CSQUTIL program
DFHMASMI	JCL program fragment that provides the WMQ definitions
DFHMASMP	JCL program fragment that provides the WMQ program compilation
DFHMASMQ	JCL program fragment that provides the WMQ definitions for the simulator
DFHMAXCV	Template for conversion

Running the installation verification procedure

After you have installed and set up the CICS Service Flow Runtime, you can use the supplied installation verification procedure (IVP) to confirm that the runtime environment is operational.

The IVP checks the configuration of the runtime environment by using pregenerated flows and the respective adapters for each type that are supported. You can select the appropriate adapters for your runtime environment using a simple interface. This provides you with the status of the IVP as it checks each adapter, and reports if the IVP fails for any reason.

The IVP uses sample programs, files, and transactions, that are optionally copied to and configured in the runtime libraries, depending on the parameters that you used when setting up the runtime environment. This section outlines the steps for ensuring that the necessary IVP resources are in place, as well as how to run the IVP from a CICS terminal.

As an optional step, you can also use the provided simulator program to initiate the business transactions that simulate CICS Service Flow Runtime processing. A positive response is another indication that the CICS Service Flow Runtime is properly installed.

The IVP instructions are documented in the following sections:

- **"Prerequisites for running the IVP"**. This section lists the prerequisites for running the IVP.
- **"Starting the IVP" on page 35**. This section tells you how to execute the IVP.
- **"Using the Simulator program to verify the installation" on page 36**. Use the instructions in this section to execute the Simulator program.
- **"IVP sample listing" on page 318**. This reference section lists the samples (programs, copybooks, maps, JCL and resource definitions) that the IVP uses.

Prerequisites for running the IVP

Before you being to run the IVP, perform the following steps.

1. If you are running the IVP for the FEPI adapter, check that the parameter **HLSYSID** has been specified in DFHMAINJ. This parameter is optional for the installation of the product, but required when running the IVP on a FEPI adapter. The adapter has also been modeled to work in a CICS region where the following conditions are met:
   - **SEC**=YES, **GMTRAN**=CESN system initialization parameters are specified.
   - RACF pass tickets are enabled for the CICS APPLID and the user ID that runs the IVP transaction

   You cannot run the IVP for FEPI adapters that use FEPI transactions, such as CZUU. These transactions are unsupported in the runtime environment.
2. Run DFHMASIM. The job is located in the runtime library SCIZSAMP and was customized with the appropriate values by DFHMAINJ. The job executes these steps:
   - Assembles and links the mapset for the sample back end programs.
   - Compiles programs DFHMABP1 through to DFHMABP5.
   - Creates WMQ programs DFHMABP6 and DFHMABP7 for the simulator. This only occurs if you specified `MQ YES` in DFHMAINJ.
   - Compiles program DFHMABP8.
   - Creates and initializes files for the back end sample program
   - Compiles simulator programs DFHMAIP1 through to DFHMAIP6.
   - Assembles and links the mapset for the simulator.
   - Creates and initializes simulator files.
   - Creates alternate indexes for the simulator files.
   - Executes DFHMASMI. This only occurs if you specified `MQ YES` in DFHMAINJ.
   - Updates the CICS CSD with the simulator definitions.
   - Updates the properties file DFHMAMPF.
3. Check the job output. If you specified `MQ YES`, check the output from the MQDEFS step. CSQUTIL can return `RC=0`, even if errors occurred. All other steps should complete successfully.

   If you get a return code of 4 and error messages IGYDS0001-W or IGYPG3112-W for certain steps, this is acceptable. If you have specified compiler options `OPTIMIZE(STD)` or `OPTIMIZE(FULL)`, you might also get a return code of 4 and error messages IGYOP3091-W, IGYPG3013-W, IGYPG3045-W, IGYPG3068-W, and IGYPG3173-W for certain steps. This is also acceptable.

The following programs are compiled:

- DFHMAB1
- DFHMABP1
- DFHMABP3
- DFHMABP4
- DFHMABP5
- DFHMABP6
- DFHMABP7
- DFHMABP8
- DFHMAIP1
- DFHMASP2
- DFHMASP4
- DFHMASP5

- DFHMASP6

When DFHMASIM completes successfully, the necessary programs, files and resources have been set up and configured for you to run the IVP.

Perform a cold restart of the CICS region. This brings the changes that were made to include the runtime libraries into effect.

Starting the IVP

Use the supplied transaction CMAA to invoke the IVP program DFHMAIVP. This transaction uses a BMS interface that allows you to select the Adapters that you want to check.

1. Start the CMAA transaction from a CICS terminal. You are presented with the following introductory menu:

```
                          CICS SFR IVP Menu

Change the default selection if required and press enter key to start IVP

   Adapter                            Selection Result
1  Link3270 Bridge server adapter ==> /
2  FEPI server adapter            ==> /
3  DPL server adapter             ==> /
4  WMQ server adapter             ==> /




PF 3 Exit
```

2. If you want to run the IVP for all the supported adapters, accept the defaults. Otherwise, remove the / for the adapter you do not want to check.
3. Press Enter when you have selected the appropriate adapters to start the IVP. When the IVP is running, the menu displays the progress for each adapter.

```
                          CICS SFR IVP Menu

Change the default selection if required and press enter key to start IVP
   Adapter                            Selection Result
1  Link3270 Bridge server adapter ==> /         Completed successfully
2  FEPI server adapter            ==> /         Running
3  DPL server adapter             ==> /         Waiting to start
4  WMQ server adapter             ==>           Not selected









Wait – IVP in progress
PF 3 Exit
```

   When the IVP has finished running, the status for each of the selected adapters should show `Completed successfully`. If an adapter fails for any reason, the IVP returns a status of `Failed`. See “Responding to IVP errors” on page 36.
4. Check the transient data queue (TDQ) CSMT to confirm that the IVP ran successfully.

5. Exit the IVP using the PF3 key.

Responding to IVP errors

If an adapter fails during the IVP, the IVP stops processing and records errors in the log, before moving on to the next adapter.

1. Check the transient data queue (TDQ) CSMT for an error message to find out why the adapter failed. If CICS cannot write to CSMT, the message is redirected to the CICS job log.
2. Follow the user response recommendations that are outlined in the error messages. For a list of IVP messages see “The IVP messages” on page 253.
3. Rerun the IVP, selecting only the adapter that failed.
4. If the adapter still fails, rerun the IVP against the adapter that has failed with auxiliary tracing switched on.
   a. In the CICS-supplied transaction CETR, select the master system and master user trace flags.
   b. Print the trace output and look for ('64'x). In particular look for any exceptions.
5. Dump the error file DFHMAERF to find out what error messages have been reported in the runtime environment.
   a. Check that the sample module DFHMAEUP is located in the *hlq*.SCIZSAMP library and has been compiled.
   b. Dump the error file using the provided sample JCL job DFHMAMED. The job is also located in the *hlq*.SCIZSAMP library.
   c. Look up the error messages that have been reported in the “Error messages” on page 220 section to see if you can take any actions to resolve the problem.

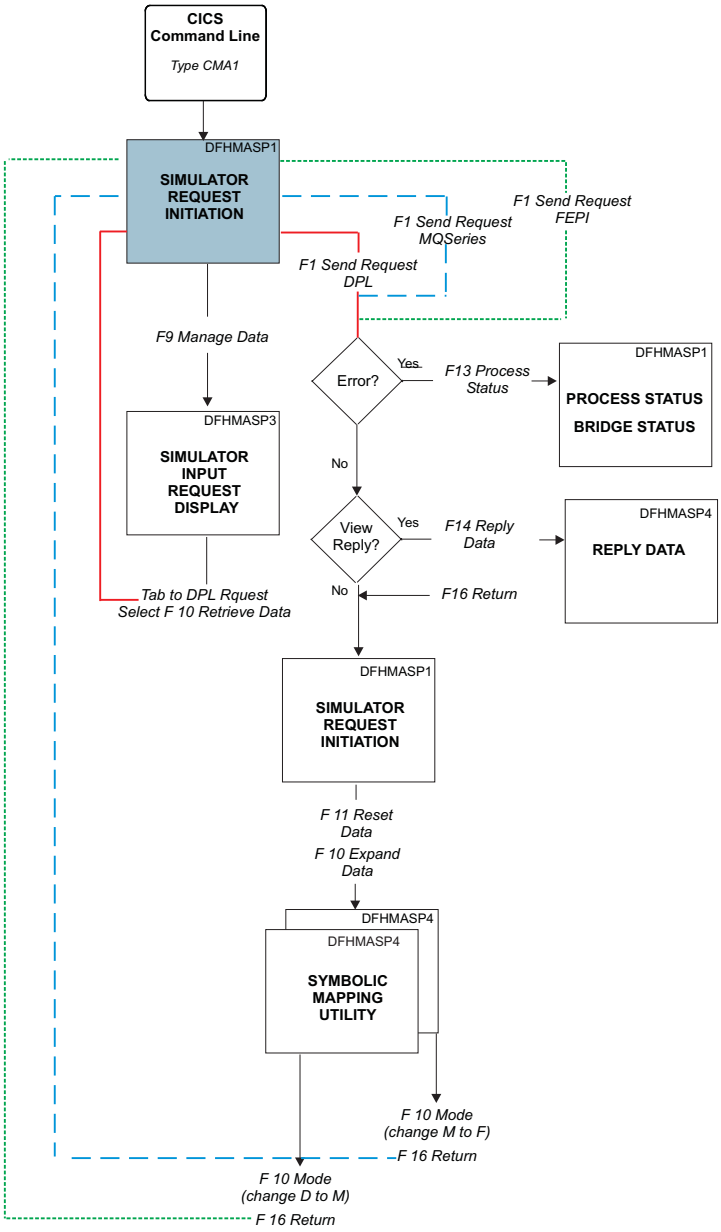
If you can't resolve the problem, contact IBM Software Support. You need to provide the trace output and the dump of the error file. It is also recommended that you provide a dump of the properties file. This is explained in “Dumping the properties file” on page 99.

Using the Simulator program to verify the installation

The IVP includes an optional Simulator program. This can also be used to check each type of adapter. The Simulator simulates the four types of adapter requests and the back-end transactions associated with each request.

In order to use the Simulator program, the runtime environment must be configured to support WebSphere MQ. Ensure that you set the parameter `MQ YES` when you ran DFHMAINJ.

The following illustration shows a high-level overview of a flow using the Simulator.



Legend:
= DPL Request
= MQSeries Request
= FEPI Request

Figure 3. IVP high-level overview

You might need to modify the initial Simulator program, DFHMASP1, or modify its display data. In addition, the IVP sample DFHMAIP1 is the sample adapter Navigator that initiates the other IVP sample server adapter programs (i.e., DFHMAIP2, DFHMAIP3, DFHMAIP4, DFHMAIP5 and DFHMAIP6). If these program names or their associated transaction IDs have been changed, you must modify DFHMAIP1 to reflect these changes as well as the CICS CSD update sample, DFHMAID1. For more information about the Simulator, see “The Simulator sample program” on page 38.

The Simulator sample program

The Simulator formats and submits a request message, in order to simulate a service requestor that is requesting services of the CICS Service Flow Runtime software.

See Figure 4 for an illustration of the processing that takes place when you issue a request using the Simulator.

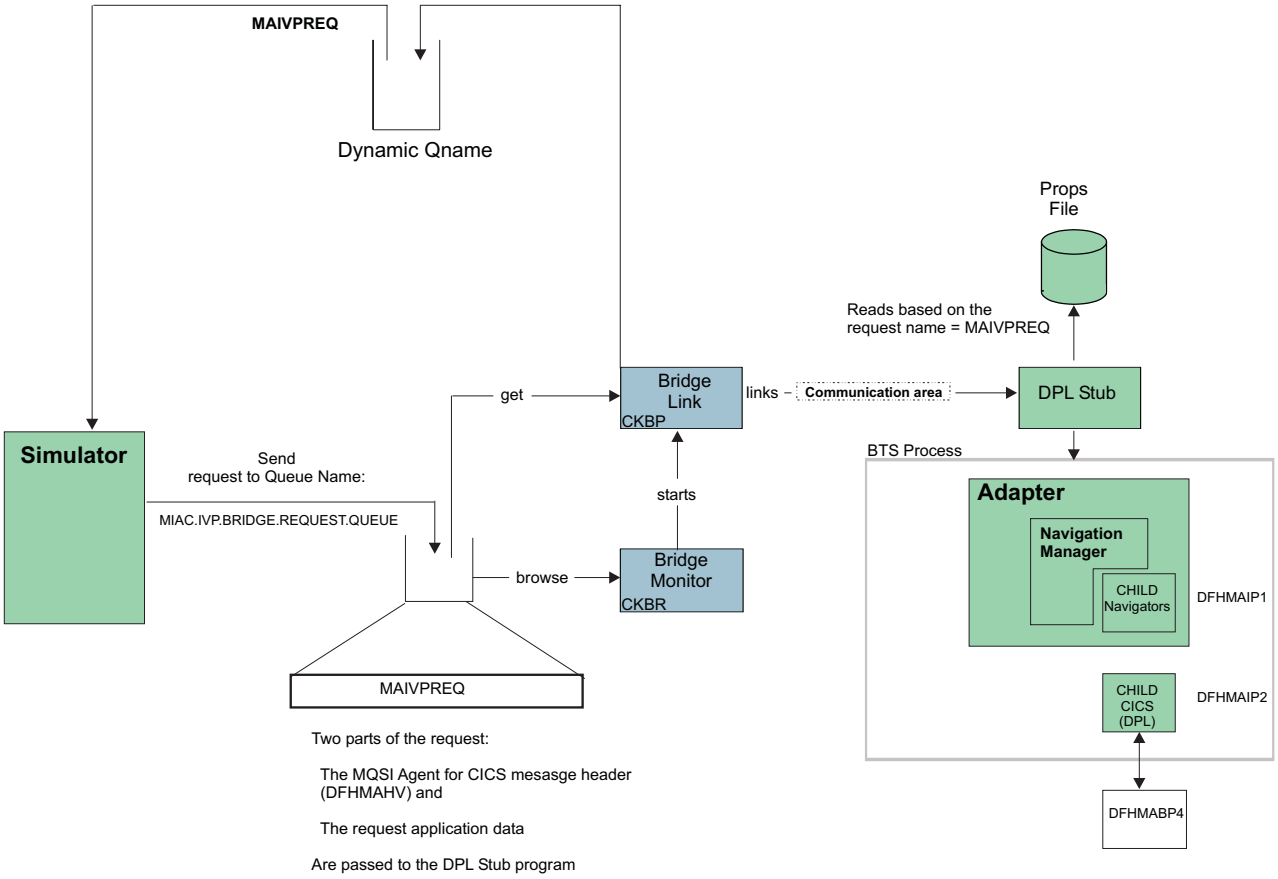


Figure 4. Request processing using the Simulator

Upon receipt of the request message from the Simulator, the server adapter that is invoked performs adapter request processing that results in a reply message. A successful reply (GOOD RESPONSE RECEIVED) is an indication that the CICS Service Flow Runtime software was properly installed and configured.

Before you use the Simulator

Before you use the Simulator you should be aware of the following conditions:

- The instructions on how to run the Simulator include the steps on how to execute all 4 IVP verification tests in succession. However, it is possible during the IVP to alter the first byte of the selected input request message in order to run the verification test that you want.
  - If the first byte in the request message is a D, the request will be used to simulate DPL server adapter processing.
  - If the first byte in the request message is an M, the request will be used to simulate WebSphere MQ server adapter processing.

  - If the first byte in the request message is an F, the request will be used to simulate FEPI server adapter processing.
  - If the first byte in the request message is an L, the request will be used to simulate Link3270 server adapter processing.
- You may need to update and compile the FEPI server adapter program (DFHMAIP5) to work within your environment. For example, if your site has a different sign-on screen, you would need to update and recompile DFHMAIP5 to accommodate this difference.
- The IVP can be initiated through either a DPL or a WebSphere MQ-CICS bridge for the interface to the CICS Service Flow Runtime DPL Stub program.

Running the Simulator

Perform the following steps to run the Simulator.

1. Start the Simulator.
2. Select the DPL request record.
3. Simulate DPL adapter request processing.
4. Check the reply to the DPL request.
5. Simulate WebSphere MQ server adapter request processing.
6. Check the reply to the WebSphere MQ server adapter request.
7. Simulate FEPI server adapter request processing.
8. Check the reply to the FEPI server adapter request. To view the reply data
9. Simulate Link3270 server adapter request processing.
10. Check the reply to the Link3270 server adapter request.

Start the Simulator:

1. From the CICS command line type `CMA1` and press Enter. CMA1 is the transaction identifier for the Simulator. The *Simulator Request Initiation* screen displays.

```
                        SIMULATOR REQUEST INITIATION               DFHMASP1
--------------------------------------------------------------------------------
 INITIATE VIA:  D (D) DPL; (M) WMQ-CICS bridge     SYSID:      if (D)PL
REQUEST QUEUE:  CIA.IVP.BRIDGE.REQUEST.QUEUE
 REQUEST NAME:  MAIVPREQ     TYPE:  1 (0) Async; (1) Sync; (2) Sync/Rollback
  PROCESSTYPE:  DFHMAINA
 PROCESS NAME:
FAIL PROCTYPE:
FAIL PROCNAME:
 REPLY QMODEL:  CIA.IVP.REPLY.MODEL.QUEUE
DYNAMIC QNAME:  SIMULATION.*
WAIT ON REPLY:  030  seconds
       USERID:                  STATE TOKEN:
 USER DATALEN:  00000        SYMBOLICS NAME:  MAIVPREQ
 MAX REPLYLEN:  00400

  Please select system option:

 (1) SEND REQUEST    (2) REPLAY DATA     (6) COMP. REQUEST   (7) CANCEL REQUEST
 (9) MANAGE DATA    (10) EXPAND DATA    (11) RESET DATA     (12) EXIT
(13) PROCESS STATUS (14) REPLY DATA                         (16) RETURN
```

Figure 5. Simulator request initiation screen — Initial appearance

2. Accept all default values. For an explanation of the fields, see “Simulator request initiation fields.”

Simulator request initiation fields:

A description of the fields that are available when you start the Simulator.

The program is DFHMASP1, which is indicated at the top of the screen. The field descriptions are as follows:

INITIATE VIA:
X(01). Choose the method by which you will initiate the IVP. Valid values include:

- D = Initiate the IVP through a DPL.
- M = Initiate the IVP through the WebSphere MQ-CICS bridge.

Identifies the queue on which the service requestor sends an adapter request message. This causes WebSphere MQ to send a trigger message to the initiation queue as defined. The WebSphere MQ trigger monitor program starts the CICS bridge monitor task, part of the WebSphere MQ-CICS bridge.

SYSID:
9(4). For DPL initialization only.

If the simulator is installed in the same region as the CICS Service Flow Runtime server run-time, leave this field blank.

If the simulator is not installed in the same region as the CICS Service Flow Runtime server run-time, type in the SYSID of the CICS Service Flow Runtime server run-time.

REQUEST QUEUE:
X(48). Identifies the queue on which the service requestor sends an adapter request message. This causes WebSphere MQ to send a trigger message to the initiation queue as defined. The WebSphere MQ trigger monitor program starts the CICS bridge monitor task, part of the WebSphere MQ-CICS bridge.

REQUEST NAME:
X(8). The request name indicates the name of the request to process. The value specified is used to read the CICS Service Flow Runtime Properties file to determine processing flow and parameters.

TYPE: X(1). Indicates the processing mode. Valid entries are:

- 0 = request processing will be run asynchronously. For requests that will be updating information when adapter request processing runs in asynchronous mode, syncpoints are taken while the Navigation Manager, Navigator and server adapters complete their processing. These synchronization points provide the necessary state, status and journaling information in the event of subsequent failure. This information can in turn be used in a compensation flow.
- 1 = **This type must be used unless modifications have been made to DFHMAID3** to change the request type on TYPE=R RECORD. A 1 indicates that the request processing will be run synchronously. For requests that will be making inquiries only (i.e., the request will not result in information being updated).
- 2 = request processing will include rollback processing. Synchronous Rollback is a processing mode where the CICS Service Flow Runtime BTS process and all activities run within the process are initiated and run in synchronous mode (i.e., BTS RUN ACQPROCESS SYNCHRONOUS and RUN ACTIVITY( ) SYNCHRONOUS commands), however, any failure within any activity within the process results in an abend of the process. This has the effect of returning the state of any and all

recoverable resources updated during adapter request processing to its original state, that is, the state prior to the execution of the failed adapter request or process.

PROCESSTYPE:
X(8). This field indicates the type of the new CICS Service Flow Runtime process instance.

FAIL PROCTYPE:
X(8). For Asynchronous mode (type 0) only. If the request fails, this field is populated with the failed process type. You can cancel the failed process by selecting (7) CANCEL REQUEST, or if the request has a compensating flow associated with it, you can select (6) COMP. REQUEST to invoke compensation. This field is also populated for *informational purposes only* when the process fails in synchronous mode. For detailed information on compensation, see “How compensation processing works” on page 193. Also, see “DFHMAH field definitions” on page 81 for information on DFHMAH-UOWCONTROL.

FAIL PROCNAME:
X(36). For Asynchronous mode (type 0) only. If the request fails, this field is populated with the failed process name. You can cancel the failed process by selecting (7) CANCEL REQUEST, or if the request has a compensating flow associated with it, you can select (6) COMP. REQUEST to invoke compensation. This field is also populated for *informational purposes only* when the process fails in synchronous mode. For detailed information on compensation, see “How compensation processing works” on page 193. Also, see “DFHMAH field definitions” on page 81 for information on DFHMAH-UOWCONTROL.

REPLY QMODEL:
X(48). Name of the model reply queue.

DYNAMIC QNAME:
X(48). Prefix used for the name of the Reply Queue when dynamically built.

WAIT INTERVAL:
9(4). Identifies how long the Simulator waits for a response. The default value is 30 seconds.

USERID:
X(8). Valid CICS UserID. This is used for authentication in WebSphere MQ-CICS bridge and to allocate a PassTicket in the FEPI server adapter.

Select the DPL request record:

1. From the *Simulator Request Initiation* screen, selection option `F9 MANAGE DATA`. The *Simulator Input Request Display* screen appears:

```
                      SIMULATOR INPUT REQUEST DISPLAY             DFHMASP3
--------------------------------------------------------------------------------

 REQUEST ID:  MAIVPREQ

 Please select system option:

  REQUEST ID     MAP          LEN       DESCRIPTION
  MAIVPREQ       MAIVPREQ      22       IVP DPL REQUEST RECORD









(1)  TOP        (2)  ADD       (3) UPDATE     (7) BACKWARD  (8) FORWARD
(10) RETRIEVE DATA           (12) EXIT      (16) RETURN    (ENTER) GO TO
```

Figure 6. Simulator input request display screen — Selecting the DPL request record

2. Tab to the IVP DPL Request Record.
3. Select F10 RETRIEVE DATA to retrieve DPL request data from the Input Request Queue. You are returned to the Simulator Request Initiation screen.

   You can proceed with this step to test DPL request processing or you can :

   - Skip to “Simulate WebSphere MQ server adapter request processing” on page 43 to test WebSphere MQ request processing.
   - Skip to “Simulate FEPI server adapter request processing” on page 46 to test FEPI request processing.
   - Skip to “Simulate Link3270 server adapter request processing” on page 48 to test Link3270 request processing.

Simulate DPL adapter request processing:

From the Simulator Request Initiation screen, select F1 SEND REQUEST to send the DPL request.

```
                        SIMULATOR REQUEST INITIATION               DFHMASP1
--------------------------------------------------------------------------------
 INITIATE VIA:  D (D) DPL; (M) WMQ-CICS bridge     SYSID:      if (D)PL
REQUEST QUEUE:  CIA.IVP.BRIDGE.REQUEST.QUEUE
 REQUEST NAME:  MAIVPREQ     TYPE:  1 (0) Async; (1) Sync; (2) Sync/Rollback
  PROCESSTYPE:  DFHMAINA
 PROCESS NAME:
FAIL PROCTYPE:
FAIL PROCNAME:
 REPLY QMODEL:  CIA.IVP.REPLY.MODEL.QUEUE
DYNAMIC QNAME:  SIMULATION.*
WAIT ON REPLY:  030  seconds
       USERID:                  STATE TOKEN:
 USER DATALEN:  00000        SYMBOLICS NAME:  MAIVPREQ
 MAX REPLYLEN:  00400

  Please select system option:

 (1) SEND REQUEST    (2) REPLAY DATA     (6) COMP. REQUEST   (7) CANCEL REQUEST
 (9) MANAGE DATA    (10) EXPAND DATA    (11) RESET DATA     (12) EXIT
(13) PROCESS STATUS (14) REPLY DATA                         (16) RETURN
```

Figure 7. Simulator request initiation screen — Sending the DPL request

If the request is processed, you receive a GOOD RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen.
If the request is not processed, you receive an ERROR RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen. See “Responding to errors in the Simulator” on page 50 for information on how to respond to this message.
For instructions on how to simulate WebSphere MQ server adapter request processing, proceed to “Simulate WebSphere MQ server adapter request processing.”

Check the reply to the DPL request:

1. To view the reply data select F14 REPLY DATA from the Simulator Request Initiation screen. The Simulator Symbolic Mapping Utility screen appears:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                 COLUMN:     1    CHAPTER: MAIVPREQ
 OFFSET:                             LEFT/RIGHT BY:    35       SIZE:    22
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: BROWSE
Please select system option:

 OFFSET NAME                           ----+----+----+----+----+----+----+----+
      0 IVP PROGRAM INDICATOR          D
      1 IVP CUSTOMER NUMBER            10000
      6 IVP USER ID
     14 IVP USER PASSWORD









 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
END OF LIST
```

Figure 8. Simulator symbolic mapping utility screen

   See “Test data for IVP back-end transactions” on page 321 to view the customer data in its entirety.
2. Select F 13 HEX to view the contents of the reply data.
3. Select F 16 to return to the Simulator Request Initiation screen.

Simulate WebSphere MQ server adapter request processing:

From the Simulator Request Initiation screen, perform the following steps:

```
                      SIMULATOR REQUEST INITIATION               DFHMASP1
--------------------------------------------------------------------------------
 INITIATE VIA:  D (D) DPL; (M) WMQ-CICS bridge     SYSID:      if (D)PL
REQUEST QUEUE:  CIA.IVP.BRIDGE.REQUEST.QUEUE
 REQUEST NAME:  MAIVPREQ     TYPE:  1 (0) Async; (1) Sync; (2) Sync/Rollback
  PROCESSTYPE:  DFHMAINA
 PROCESS NAME:
FAIL PROCTYPE:
FAIL PROCNAME:
 REPLY QMODEL:  CIA.IVP.REPLY.MODEL.QUEUE
DYNAMIC QNAME:  SIMULATION.*
WAIT ON REPLY:  030  seconds
       USERID:                  STATE TOKEN:
 USER DATALEN:  00000        SYMBOLICS NAME:  MAIVPREQ
 MAX REPLYLEN:  00400

  Please select system option:

 (1) SEND REQUEST    (2) REPLAY DATA     (6) COMP. REQUEST   (7) CANCEL REQUEST
 (9) MANAGE DATA    (10) EXPAND DATA    (11) RESET DATA     (12) EXIT
(13) PROCESS STATUS (14) REPLY DATA                         (16) RETURN
```

Figure 9. Simulator Request Initiation screen — Resetting data for an WebSphere MQ server adapter request

1. Select F11 RESET DATA. The Process name disappears.
2. Select F10 EXPAND DATA. The Simulator Symbolic Mapping Utility screen displays in **BROWSE** mode:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                  COLUMN:     1    CHAPTER: MAIVPREQ
 OFFSET:                              LEFT/RIGHT BY:    35       SIZE:    22
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: BROWSE
Please select system option:

 OFFSET NAME                           ----+----+----+----+----+----+----+----+
      0 IVP PROGRAM INDICATOR          D
      1 IVP CUSTOMER NUMBER            10000
      6 IVP USER ID
     14 IVP USER PASSWORD









 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
END OF LIST
```

Figure 10. Simulator symbolic mapping utility screen — Browse mode

3. Select F10 MODE to change from BROWSE mode to **UPDATE** mode.
4. Tab to **D** and change the D to an M. An M indicates that you want to simulate WebSphere MQ adapter request processing.
5. Press Enter. The screen should appear like the following when you are through:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                    COLUMN:     1    CHAPTER: MAIVPREQ
 OFFSET:                                LEFT/RIGHT BY:    35       SIZE:    22
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: UPDATE
Please select system option:

 OFFSET NAME                             ----+----+----+----+----+----+----+
      0 IVP PROGRAM INDICATOR            M
      1 IVP CUSTOMER NUMBER              10000
      6 IVP USER ID
     14 IVP USER PASSWORD




 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
END OF LIST
```

Figure 11. Simulator symbolic mapping utility screen — Changing to WebSphere MQ server adapter processing

6. Select F16 RETURN to go back to the Simulator Request Initiation screen.
7. Select F1 SEND REQUEST to simulate WebSphere MQ server adapter request processing.

   If the request is processed, you receive a GOOD RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen.

   If the request is not processed, you receive an ERROR RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen. See “Responding to errors in the Simulator” on page 50 for information on how to respond to this message.

Check the reply to the WebSphere MQ server adapter request.:

To view the reply data:

1. Select F14 REPLY DATA from the Simulator Request Initiation screen. The Simulator Symbolic Mapping Utility screen appears:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                  COLUMN:     1    CHAPTER:
 OFFSET:                               LEFT/RIGHT BY:    35       SIZE:   400
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: BROWSE
Please select system option:

 OFFSET NAME                            ----+----+----+----+----+----+----+
      0 FILLER                          10000ONE           CUSTOMER









 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
SYMBOLICS WERE NOT FOUND FOR AREA SELECTED - FILLER USED TO MAP TEMP STORAGE
```

Figure 12. Simulator symbolic mapping utility screen — View reply to WebSphere MQ server adapter request

2. Select `F 13 HEX` to view the contents of the reply data.
3. Select `F 16` to return to the Simulator Request Initiation screen.

Simulate FEPI server adapter request processing:

From the Simulator Request Initiation screen, perform the following steps:

```
                        SIMULATOR REQUEST INITIATION               DFHMASP1
--------------------------------------------------------------------------------
 INITIATE VIA:  D (D) DPL; (M) WMQ-CICS bridge     SYSID:      if (D)PL
REQUEST QUEUE:  CIA.IVP.BRIDGE.REQUEST.QUEUE
 REQUEST NAME:  MAIVPREQ     TYPE:  1 (0) Async; (1) Sync; (2) Sync/Rollback
  PROCESSTYPE:  DFHMAINA
 PROCESS NAME:
FAIL PROCTYPE:
FAIL PROCNAME:
 REPLY QMODEL:  CIA.IVP.REPLY.MODEL.QUEUE
DYNAMIC QNAME:  SIMULATION.*
WAIT ON REPLY:  030  seconds
       USERID:                  STATE TOKEN:
 USER DATALEN:  00000        SYMBOLICS NAME:  MAIVPREQ
 MAX REPLYLEN:  00400

  Please select system option:

 (1) SEND REQUEST    (2) REPLAY DATA     (6) COMP. REQUEST   (7) CANCEL REQUEST
 (9) MANAGE DATA    (10) EXPAND DATA    (11) RESET DATA     (12) EXIT
(13) PROCESS STATUS (14) REPLY DATA                         (16) RETURN
```

Figure 13. Simulator Request Initiation screen — Resetting data for a FEPI server adapter request

1. Select `F11 RESET DATA`.
2. Select `F10 EXPAND DATA`. The Simulator Symbolic Mapping Utility screen displays in Browse mode:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                 COLUMN:     1    CHAPTER: MAIVPREQ
 OFFSET:                              LEFT/RIGHT BY:    35       SIZE:    22
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: BROWSE
Please select system option:

 OFFSET NAME                           ----+----+----+----+----+----+----+
      0 IVP PROGRAM INDICATOR          M
      1 IVP CUSTOMER NUMBER            10000
      6 IVP USER ID
     14 IVP USER PASSWORD




 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
END OF LIST
```

Figure 14. Simulator symbolic mapping utility screen — MQ record

3. Select F10 MODE to change from BROWSE mode to UPDATE mode.
4. Tab to **M** and change the M to F. An F indicates that you will simulate FEPI adapter request processing.
5. Press Enter. The screen should look like this when you are through:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                 COLUMN:     1    CHAPTER: MAIVPREQ
 OFFSET:                              LEFT/RIGHT BY:    35       SIZE:    22
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: UPDATE
Please select system option:

 OFFSET NAME                           ----+----+----+----+----+----+----+
      0 IVP PROGRAM INDICATOR          F
      1 IVP CUSTOMER NUMBER            10000
      6 IVP USER ID
     14 IVP USER PASSWORD




 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
END OF LIST
```

Figure 15. Simulator symbolic mapping utility screen — Changing to FEPI server adapter processing

6. Select F16 RETURN to go back to the Simulator Request Initiation screen.
7. Select F1 SEND REQUEST to simulate FEPI server adapter request processing.

   If the request is processed, you receive a GOOD RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen.

   If the request is not processed, you receive a ERROR RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen. See “Responding to errors in the Simulator” on page 50 for information on how to respond to this message.

Check the reply to the FEPI server adapter request:

To view the reply data:

1. Select F14 REPLY DATA from the Simulator Request Initiation screen. The Simulator Symbolic Mapping Utility screen appears:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                  COLUMN:     1    CHAPTER:
 OFFSET:                             LEFT/RIGHT BY:    35       SIZE:   400
LIST BY: O (O)FFSET/(N)AME/(S)EQ      UPDATE MODE: BROWSE
Please select system option:

 OFFSET NAME                            ----+----+----+----+----+----+----+----+
      0 FILLER                          10000ONE           CUSTOMER









 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
SYMBOLICS WERE NOT FOUND FOR AREA SELECTED - FILLER USED TO MAP TEMP STORAGE
```

Figure 16. Simulator symbolic mapping utility screen — View reply to FEPI server adapter request

2. Select F 13 HEX to view the contents of the reply data.
3. Select F 16 to return to the Simulator Request Initiation screen.

Simulate Link3270 server adapter request processing:
From the Simulator Request Initiation screen, perform the following steps:

```
                        SIMULATOR REQUEST INITIATION                DFHMASP1
--------------------------------------------------------------------------------
 INITIATE VIA:  D (D) DPL; (M) WMQ-CICS bridge     SYSID:      if (D)PL
REQUEST QUEUE:  CIA.IVP.BRIDGE.REQUEST.QUEUE
 REQUEST NAME:  MAIVPREQ     TYPE:  1 (0) Async; (1) Sync; (2) Sync/Rollback
  PROCESSTYPE:  DFHMAINA
 PROCESS NAME:
FAIL PROCTYPE:
FAIL PROCNAME:
 REPLY QMODEL:  CIA.IVP.REPLY.MODEL.QUEUE
DYNAMIC QNAME:  SIMULATION.*
WAIT ON REPLY:  030  seconds
       USERID:                  STATE TOKEN:
 USER DATALEN:  00000        SYMBOLICS NAME:  MAIVPREQ
 MAX REPLYLEN:  00400

  Please select system option:

 (1) SEND REQUEST    (2) REPLAY DATA     (6) COMP. REQUEST   (7) CANCEL REQUEST
 (9) MANAGE DATA    (10) EXPAND DATA    (11) RESET DATA     (12) EXIT
(13) PROCESS STATUS (14) REPLY DATA                         (16) RETURN
```

Figure 17. Simulator Request Initiation screen — Resetting data for a Link3270 server adapter request

1. Select F11 RESET DATA.

2. Select F10 EXPAND DATA. The Simulator Symbolic Mapping Utility screen displays in Browse mode:

```
                          SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                  COLUMN:     1    CHAPTER: MAIVPREQ
 OFFSET:                               LEFT/RIGHT BY:    35       SIZE:    22
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: BROWSE
Please select system option:

 OFFSET NAME                            ----+----+----+----+----+----+----+
      0 IVP PROGRAM INDICATOR           F
      1 IVP CUSTOMER NUMBER             10000
      6 IVP USER ID
     14 IVP USER PASSWORD




 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
END OF LIST
```

Figure 18. Simulator symbolic mapping utility screen — FEPI record

3. Select F10 MODE to change from BROWSE mode to UPDATE mode.
4. Tab to **F** and change the F to L. An L indicates that you will simulate Link3270 bridge server adapter request processing.
5. Press Enter. The screen should look like this when you are through:

```
                          SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                  COLUMN:     1    CHAPTER: MAIVPREQ
 OFFSET:                               LEFT/RIGHT BY:    35       SIZE:    22
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: UPDATE
Please select system option:

 OFFSET NAME                            ----+----+----+----+----+----+----+
      0 IVP PROGRAM INDICATOR           L
      1 IVP CUSTOMER NUMBER             10000
      6 IVP USER ID
     14 IVP USER PASSWORD




 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
END OF LIST
```

Figure 19. Simulator symbolic mapping utility screen — Changing to Link370 server adapter processing

6. Select F16 RETURN to go back to the Simulator Request Initiation screen.
7. Select F1 SEND REQUEST to simulate Link3270 server adapter request processing.

   If the request is processed, you receive a GOOD RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen.

If the request is not processed, you receive a ERROR RESPONSE RECEIVED in the message area of the Simulator Request Initiation screen. See “Responding to errors in the Simulator” for information on how to respond to this message.

Check the reply to the Link3270 server adapter request:
To view the reply data:

1. Select F14 REPLY DATA from the Simulator Request Initiation screen.

   The Simulator Symbolic Mapping Utility screen appears:

```
                        SIMULATOR SYMBOLIC MAPPING UTILITY         DFHMASP4
--------------------------------------------------------------------------------


   NAME:                                  COLUMN:     1    CHAPTER:
 OFFSET:                               LEFT/RIGHT BY:    35       SIZE:   400
LIST BY: O (O)FFSET/(N)AME/(S)EQ     UPDATE MODE: BROWSE
Please select system option:

 OFFSET NAME                            ----+----+----+----+----+----+----+----+
      0 FILLER                          10000ONE           CUSTOMER









 (1) TOP   (3) SELECT  (4) VIEW  (7) BACKWARD  (8) FORWARD  (10) MODE (12) EXIT
(13) HEX  (16) RETURN (19) LEFT (20) RIGHT    (22) INIT     (ENTER) GO TO
SYMBOLICS WERE NOT FOUND FOR AREA SELECTED - FILLER USED TO MAP TEMP STORAGE
```

Figure 20. Simulator symbolic mapping utility screen — View reply to Link3270 server adapter request

2. Select F 13 HEX to view the contents of the reply data.
3. Select F 16 to return to the Simulator Request Initiation screen.

Responding to errors in the Simulator

There are several messages that can appear in the message area of the Simulator Request Initiation display (DFHMASP1).

These messages indicate the Simulator status or denote any errors that have occurred while using the Simulator. See “IVP simulator messages” on page 52 for a list of the errors can display in the message area of the Simulator Request Initiation display.

In response to these messages you should perform the following steps:

1. Select F 13 PROCESS STATUS. The Process Status information displays as follows:

```
                      SIMULATOR REQUEST INITIATION              DFHMASP1
------------------------------------------------------------------------------

                              PROCESS STATUS

  COMPLETION STATUS:  NORMAL
              MODE:  COMPLETE
     SUSPEND STATUS:  NOTSUSPENDED
         ABEND CODE:
        RETURN CODE:  00000009
            MESSAGE:  CIA04002
     FAILED TRANSID:  CMA9
     FAILED PROGRAM:  DFHMAIP5
        FAILED NODE:  DFHMAIP5









  Please select system option:
                                                 (12) EXIT  (16) RETURN
```

Figure 21. Process status screen

If an error occurs during WebSphere MQ-CICS bridge processing, the display fields come from the returned MQCIH header and the screen that appears when you select `F 13 PROCESS STATUS` is as follows:

```
                      SIMULATOR REQUEST INITIATION              DFHMASP1
------------------------------------------------------------------------------

                               BRIDGE STATUS

        RETURN CODE:  SECURITY ERROR
           FUNCTION:  1008
    COMPLETION CODE:  000000069
        REASON CODE:  000000008
         ABEND CODE:
```

Figure 22. WebSphere MQ-CICS bridge status screen

2. View the information on the status screen(s) to determine why the request failed. Any information (such as, return code, completion status, suspend status, mode, message) displayed on the Process Status screen comes from the DFHMAH structure. The following lists the DFHMAH fields that correspond to the display item:
   - DFHMAH-COMPSTATUS - Completion Status
   - DFHMAH-MODE - Mode
   - DFHMAH-SUSPSTATUS - Suspend Status
   - DFHMAH-ABENDCODE - Abend Code

- DFHMAH-RETURNCODE - Return Code
- DFHMAH-MESSAGE - Message
- DFHMAH-FAILED-TRANID - Failed Transid
- DFHMAH-FAILED-PROGRAMID - Failed Program
- DFHMAH-FAILED-NODE - Failed Node

See “Request message headers” on page 80 for detailed information on these fields.

If you are viewing the bridge Status screen, refer to *WebSphere MQ Application Programming Reference* for the MQCIH ReturnCode, Function, CompCode, Reason, and AbendCode definitions and values.

IVP simulator messages:

The following messages can occur when using the Simulator to run the IVP. See “Responding to errors in the Simulator” on page 50 for information on how to view error message information from the Simulator.

GOOD RESPONSE RECEIVED

Indicates a reply message was received from the IVP test programs with return code equal to zero.

ERROR RESPONSE RECEIVED

Indicates a reply message was received from the IVP test programs with return code greater than zero.

MQ PUT TO REQUEST QUEUE FAILED. REASON CODE = xxxx

Indicates the MQPUT to the specified CICS bridge request queue failed with MQPUT1 reason code = xxxx. See the *WebSphere MQ Application Programming Reference* for reason code definitions.

MQ GET FROM REPLY QUEUE FAILED. REASON CODE = xxxx

Indicates the MQGET from the Dynamic reply queue, DYNAMIC QNAME field, generated failed with MQGET reason code = xxxx. See the *WebSphere MQ Application Programming Reference* for reason code definitions.

NO RESPONSE YET!

Indicates no response has been received on the Dynamic reply queue, DYNAMIC QNAME field, within the wait interval specified, WAIT ON REPLY field.

REQUIRED FIELD(S) MUST BE ENTERED

One of the required fields has not been specified. Required fields are as follows:

- Request Queue
- Request Name
- Type
- Processtype
- Reply Qmodel
- Dynamic Qname
- Max Replylen

ERROR ON CALL TO PROGRAM xxxxxxxx

A call to another Simulator module has failed. xxxxxxxx = DFHMASP3 or DFHMASP4.

Chapter 4. Planning

There are a number of activities that you should perform when planning to use the Service Flow Modeler with CICS Service Flow Runtime

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

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

You can find out more information about developing and deploying using Service Flow Modeler in the WebSphere Developer for System 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:

- 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 pain points.
- Understand in detail the applications and systems to be transformed.

Planning phase of an application transformation project

1. Plan and design your implementation. This includes:
   - Understanding and designing business transactions.
   - Deciding if and how to implement recoverability at the request message level. See “Determine if compensation is necessary” on page 58 for things to consider when deciding whether or not to include recoverability in a request message.
   - Deciding how to implement security.
   - Determining the BTS configuration(s) necessary to support adapter request processing.
2. Compensation (journaling support)
3. Auditing and error logging
4. Developing a test that proves the modeled run time works as expected.

Deployment patterns

Developers use Service Flow Modeler to define a combination of elements or characteristics that describe how the adapter service will run in the CICS Service Flow Runtime environment to process requests.

© Copyright IBM Corp. 2001, 2010

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

Deployment patterns manifest in the modeled adapter service at build time.

Single connector *simple* patterns

A single connector pattern consists of a single generated and deployed server adapter program as modeled in Service Flow Modeler to compose an adapter service. The server adapter program can consist of one or more interactions with multiple target systems, but does not require an Adapter Navigator to manage request processing. This pattern is intended to support simple screen-sequencing if the server adapter type is FEPI or Link3270, or where a distributed programming link is invoked using a DPL server adapter. Only one server adapter can be generated in a single connector pattern.

WebSphere MQ server adapters, the DPL server adapter that is generated with version 7 of the tooling, and the Web services server adapter are not supported in a single connector pattern.

There are two single connector patterns:

Nonpersistent single connector pattern
: The term *nonpersistent* indicates that no record is written to the BTS repository data set to reserve the name of the BTS process. This is implemented using 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 BTS `NOCHECK` option is not specified on the **`BTS DEFINE PROCESS`** command. Omitting this option ensures that the context of the adapter service (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 generated and deployed server adapter programs as modeled in Service Flow Modeler to compose an adapter service. 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 adapter Navigator to mediate and control the flow of the adapter service. During adapter service request processing, multiple server adapters can be run without requiring action by the service requestor.

There are two aggregate connector patterns:

Nonpersistent single connector pattern

The term *nonpersistent* indicates that no record is written to the BTS repository data set to reserve the name of the BTS process. This is implemented using 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 BTS `NOCHECK` option is not specified on the **`BTS DEFINE PROCESS`** command. Omitting this option ensures that the context of the adapter service (the request and reply data, as well as an intermediate state data) is persistent through a failure.

Passthrough pattern

In addition to the single and aggregate patterns, a passthrough processing pattern is also supported to run target CICS transactions using the CICS Link3270 bridge mechanism only. Passthrough processing refers to the programmatic functions coded in, and managed by the service requestor in order to fulfill a business transaction using the CICS Link3270 bridge mechanism. The BTS NOCHECK option is always used for passthrough processing.

Although developers do not model or deploy passthrough processing as an adapter service in Service Flow Modeler, it is included here for clarity as it is neither a single connector nor aggregate connector pattern.

See the *CICS External Interfaces Guide Version 2 Release 2* or higher for information regarding the Link3270 bridge mechanism and related processing.

Select the processing mode to use

Requests can be processed in *synchronous mode* or in *asynchronous mode*, regardless of the deployment pattern associated with the Adapter service.

The fundamental difference between adapter request processing in asynchronous mode versus adapter request processing in synchronous mode is:

- The BTS process implementing an instance of the CICS Service Flow Runtime **for inquiry requests**, is run in the same unit of work with the same commit scope as the WebSphere MQ-CICS bridge link task. The DPL stub program and the BTS process initiated by the stub program are run synchronously as part of this single unit of work
- The BTS process implementing an instance of the CICS Service Flow Runtime **for update requests**, is run asynchronously from the initiating unit of work. All BTS activities within that BTS process are run asynchronously from their parent activities. This has the effect of running the BTS process and all activities as separate units of work, each with a distinct commit scope.

Running asynchronously means that data can only be added to the output message after the execution of the last server adapter. If your modeled flows require data to be moved to the output message between the invocation of server adapters, i.e. in mid-flow, use the synchronous processing mode. This restriction is removed when you apply APAR PK32131.

Under normal circumstances, if you are invoking the runtime environment using a CICS-supplied interface, you cannot run in asynchronous mode.

How to run your business transaction request in asynchronous mode

At build time you need to take the following steps to run a business transaction request in asynchronous mode.

1. The request type in the Properties file must be set to 0. This indicates that the request will run in asynchronous mode.
2. The service requestor should not load data into the following fields of the MQMD Message Descriptor:
   - ReplyToQ
   - ReplyToQMgr

   These two fields control the WebSphere MQ-CICS bridge program response. When you run your business transaction in asynchronous mode, the WebSphere MQ-CICS bridge cannot issue the resultant reply to the original request. This is accomplished directly from the Navigation Manager, run as DFHROOT within the BTS process.
3. The service requestor must load the ReplyToQ and ReplyToQMgr in the DFHMAH message header.
4. Optional: The service requestor can optionally load message ID MSGID and correlation ID CORRELID in the DFHMAH message header. By convention, the CORRELID of the reply message will contain the message ID of the request message.

If an error occurs before initiating a BTS process, the DPL stub program can also issue replies in asynchronous mode.

Considering security

There are two ways in which security can be implemented

- The adapter treats the service requestor as a trusted link. The service requestor takes full responsibility for security.
- The adapter does not treat the service requestor as a trusted link. In this case, security must be implemented in the runtime environment (i.e., CICS).

Security can be implemented in the CICS Service Flow Runtime using existing facilities in the following components:

- WebSphere MQ-CICS bridge
- CICS RACF, or other external security manager. See the*CICS RACF Security Guide* for information on RACF.
- CICS FEPI

CICS Service Flow Runtime does not require users to sign on before issuing requests for processing. However, you can specify that authentication levels are checked based on the userid or password or both in request messages for CICS programs that are run as part of the runtime environment.

Using IBM WebSphere MQ-CICS bridge to control authentication

The WebSphere MQ-CICS bridge is the control point for establishing the authentication level required.

Application programmers who want to code the service requestor to use the WebSphere MQ-CICS bridge, should see the *WebSphere MQ Application Programming Guide* and the *WebSphere MQ Application Programming Reference*.

1. Using the **AUTH** parameter that is passed to the CICS bridge monitor task at startup, the authentication level required for the CICS bridge link task is established. If you are using FEPI with PassTickets, you must set the **AUTH** parameter to a value other than LOCAL.

   See the *WebSphere MQ for OS/390 V2R1 System Management Guide* for information on Security considerations while using WebSphere MQ with CICS.
2. Based on the authentication level requested and on the userid or password or both in the request message (MQMD header structure and MQCIH header structure, respectively), checks are made to ensure the user has the authority to run the particular CICS bridge link task.
3. Subsequent CICS Service Flow Runtime programs and processes will run in CICS under the authentication level that was established for the CICS bridge link task. See the *CICS Business Transaction Services* manual for information on security.

The authentication level required for the CICS bridge monitor task is controlled and determined by the mechanism used to initiate the task. See the *WebSphere MQ for OS/390 V2R1 System Management Guide* for information on security considerations for using WebSphere MQ with CICS .

To establish different levels of authentication, it is possible to initiate multiple bridge monitor tasks with different **AUTH** parameters specified. One potential scenario might have request messages with varying degrees of importance processed by separate bridge monitor tasks, each with a specific level of authority.

The service requestor would send request messages at one level of importance to one queue. The service requestor would send request messages at a different level of importance to a different queue.

When a CICS application is accessed via FEPI, a PassTicket can be requested that is used in conjunction with the original userid to sign on to the application. See the *CICS Front End Programming User's Guide* for information on how to use PassTickets .

Setting the audit level

Auditing is a function of BTS. The audit level is controlled by a BTS process type passed in the adapter request message. The audit level determines the audit points.

If enabled, auditing is implemented at run time by BTS and is based on the BTS process type passed in the adapter request message.

The audit levels that are supported, 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, see the *CICS Business Transaction Services* manual.

Determine if compensation is necessary

Compensation is the act of modifying the effects of a completed activity and can be used to undo or reverse the actions that the activity took.

At build time, users need to know the modeled service flows that require recoverability if a failure occurs at run time. If it is decided that a service flow requires recoverability, the person that models the service flow also needs to model a separate service flow that can be used for compensation.

The requirement for addressing recoverability in adapter request processing is a business decision. A basic guideline could be as follows:

- If data is updated as a result of FEPI server adapter request processing or Link3270 bridge server adapter request processing (for example in a funds transfer transaction), a compensation service flow should be provided. A request that requires compensation is processed at run time in *asynchronous mode* if modeled.

  When the process terminates abnormally in asynchronous mode (due to a runtime system error or an improperly modeled flow), the runtime BTS process is *incomplete*.
- If a request is merely inquiring on status (an account inquiry for example), a compensation service flow is not necessary. A request that does not require compensation is processed at run time in *synchronous mode*.

  In Inquiry (synchronous mode) processing, even if the process does not execute successfully, the process state is considered *complete*.

For information on what steps the service requestor can take when the BTS process is incomplete, see Table 3.

Table 3. How to handle failed process in CICS Service Flow Runtime

Mode	Process execution	Process state	Next step	End result
Asynchronous (update)	Failed	Incomplete BTS resources, including process container information and process state are available for use in subsequent CICS Service Flow Runtime processes.	The service requestor can either: • initiate a process of a modeled flow to *compensate* for the failed process or, • issue a *cancel* request to the CICS Service Flow Runtime to release BTS resources. See “Deciding on whether to cancel or run compensating flow” on page 60 for more information.	A cancel request releases the BTS resources and ends the BTS process. For information on how the compensation flow is invoked for a FEPI adapter, see “LU assignment processing for non-unique IDs — asynchronous mode” on page 162. For information on how the compensation flow is invoked for a Link3270 bridge server adapter, see “Link3270 State file processing for non-unique IDs — asynchronous mode” on page 371.

Table 3. How to handle failed process in CICS Service Flow Runtime (continued)

Mode	Process execution	Process state	Next step	End result
Asynchronous (update)	Successful	Complete No BTS resources are left outstanding. No process container information is available for use in subsequent CICS Service Flow Runtime processes.	None	No BTS resources are left outstanding. No process container information is available for use in subsequent CICS Service Flow Runtime processes.
Synchronous (inquiry)	Failed	Complete No process state or container information is available for use in subsequent CICS Service Flow Runtime processes and no BTS resources are left outstanding.	If your site uses *LU assignment processing for non-unique UserIDs*, abnormal termination can result in LUs remaining assigned (for FEPI adapter request processing) or can result in allocated Link3270 bridge facilities and associated Link3270 State file data on the CICS Service Flow Runtime Link3270 State file (for Link3270 bridge server adapter request processing). Because of the *complete* state there will be no BTS process container information available for subsequent use to locate, use and logoff assigned LUs (for FEPI adapter request processing) or to locate, use and delete bridge facilities and associated state data (for Link3270 bridge server adapter request processing).	For FEPI adapter request processing, the LUs that were assigned will have their sessions (FEPI conversations) terminated and assignment deleted before the BTS process is complete, successfully executed or not. See “LU assignment processing for non-unique IDs — synchronous mode” on page 160 for a description of how LU assignments are deleted. For Link3270 bridge server adapter request processing, the bridge facilities (and associated state data) will be deleted before the BTS process is complete, successfully executed or not. See “Link3270 State file processing for non-unique IDs — synchronous mode” on page 369 for a description of how this is accomplished.

Table 3. How to handle failed process in CICS Service Flow Runtime (continued)

Mode	Process execution	Process state	Next step	End result
Synchronous (inquiry)	Successful	Complete No process state or container information is available for use in subsequent CICS Service Flow Runtime processes and no BTS resources are left outstanding.	For FEPI adapter request processing, if FEPI LUs are left assigned for use in subsequent process execution, the service requestor is responsible to logoff and cleanup LUs for a specific userid when the use of that userid is no longer required. For Link3270 bridge server adapter request processing, if the delete of bridge facilities fails, those bridge facilities will remain until the expiration of bridge facility *maximum keeptime*, (BRIH-FACILITYKEEPTIME). For information on how to set the Link3270 bridge facility *maximum keeptime*, see information on **MAT_MAX_FAC_** **KEEPTIME** in the Service Flow Modeler help in theWebSphere Developer for System z infocenter.	As a normal end of day processing strategy, the service requestor could invoke a modeled flow as a process to locate, logoff and release any assigned LUs.

Deciding on whether to cancel or run compensating flow

The action chosen (compensate or cancel) to complete the failed BTS process is set by the service requestor and is specified in the DFHMAH header field, DFHMAH-UOWCONTROL. See “DFHMAH field definitions” on page 81 for information about DFHMAH-UOWCONTROL.

The CICS Service Flow Runtime DPL Stub program, DFHMADPL, uses the indicator in DFHMAH-UOWCONTROL to determine that the request is intended to *compensate* or *cancel* a previously failed BTS process.

The DPL Stub program also uses the DFHMAH-FAILED-PROCNAME and DFHMAH-FAILED-PROCTYPE in the DFHMAH header information to locate the failed BTS process. In so doing, the DPL stub program gains access to the container information of that failed process.

If you are using *LU assignment processing for non-unique UserIDs*, the DPL Stub program employs logic to perform additional processing.

- If you are using a FEPI adapter, see “LU assignment processing for unique IDs — asynchronous mode” on page 161 and “LU assignment processing for unique IDs — synchronous mode” on page 160 for more information.

- If you are using a Link3270 bridge server adapter, see “Link3270 State file processing for unique IDs — asynchronous mode” on page 370 and “Link3270 State file processing for unique IDs — synchronous mode” on page 368 for more information.

Chapter 5. Deploying an adapter service

When you generate an adapter service in Service Flow Modeler, a series of steps is required to deploy it to CICS.

The deployment steps can be summarized as follows:

1. Compile the output from Service Flow Modeler.
2. Define the adapter service to CICS.
3. Update the properties file.

The deployment process is shown in the following figure.

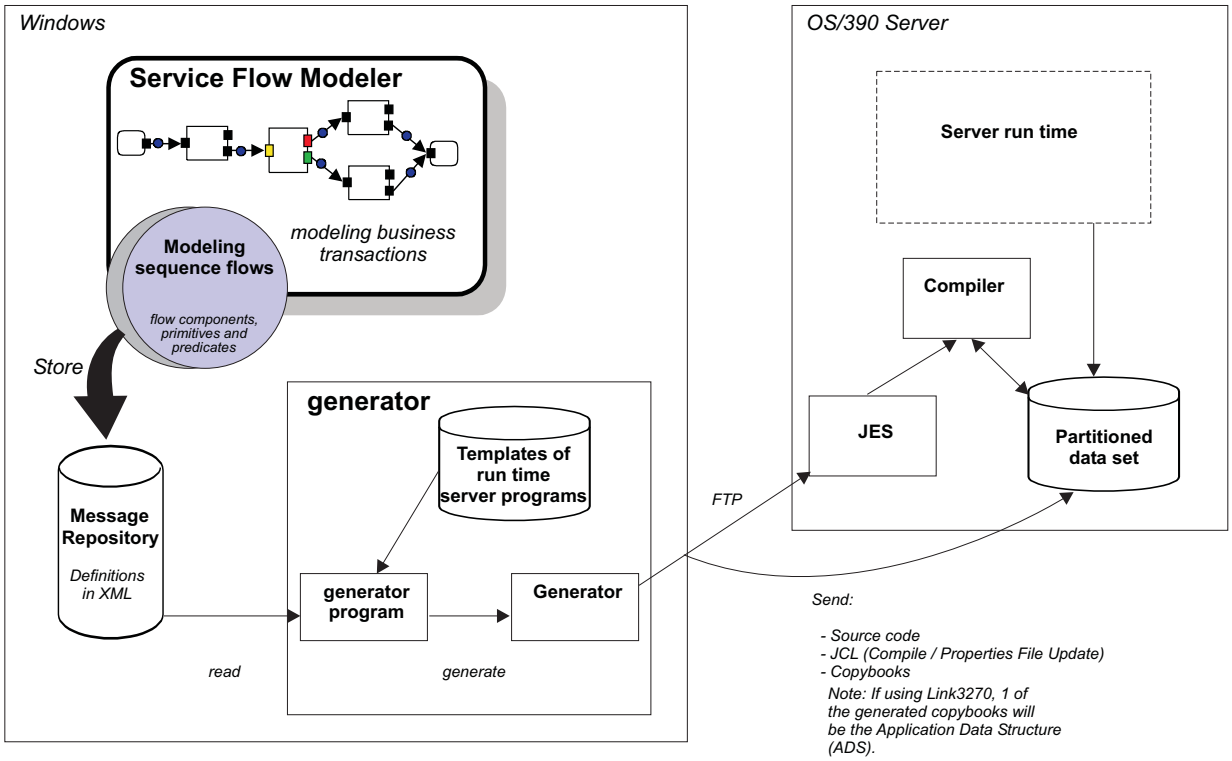


Figure 23. Deploying an adapter service from the Service Flow Modeler to the z/OS server

Deploying a new adapter service

When an adapter service is generated in the tooling, all of the necessary files are created for you to quickly deploy it into CICS. These files include JCL procedures that will compile the required programs, define the resources to CICS and update the properties file.

You can either run the JCL procedures remotely from Service Flow Modeler or run them directly in MVS. If you want to run them remotely, you must have remote systems configured in the tooling.

1. Optional: Copy the generated adapter service output from the tooling to the correct location in MVS. If you have remote systems configured and select to remotely deploy the adapter service, this will occur automatically as part of the generation process.

© Copyright IBM Corp. 2001, 2010

2. Run the batch job RDOJOB.jcl to define the programs, transactions and files to CICS. RDOJOB.jcl is created when the adapter service is generated using WebSphere Developer for System z version 6. If you are using version 7 of the tooling, the JCL is generated with the format `#jobname.jcl`, where *jobname* is the name of the service flow project. The JCL uses a CICS-supplied utility called DFHCSDUP to update the CSD of the CICS region with the necessary resource definitions. If a resource is already defined, the job does not replace it.
3. Run the batch job *@jobname*.jcl to update the properties file. If you are using version 7 of the tooling, *jobname* is the name of the request defined for the flow. A sample job called DFHMAMPU is also provided. This JCL runs the properties file update utility DFHMAMUP, that adds records for your adapter service and associated server adapters to the properties file.

   The job has a **MODE** parameter that determines whether or not the properties file should be updated if it detects a naming conflict. By default, the parameter is set to `MODE=SAFE`, to avoid overwriting any definitions of the same name in the properties file. If you get a return code other than `0` when you run the job, check the output. There could be a naming conflict to resolve. You can either change the batch job to overwrite the definitions in the properties file using `MODE=OVERWRITE`, or you can change the values in your batch job.
4. Run any remaining JCL jobs to compile the programs that are needed by your adapter service.
5. Install the CICS resources using the group name. The group must belong to the RDO list that was specified in the **RDOLIST** parameter of the post-installation procedure DFHMAINJ. The default list is CICSSFRL.
6. Optional: If your adapter service is enabled as a Web service provider, or includes an outbound Web service request:
   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 post-installation steps.
   b. Copy the generated Web service binding file and optionally the WSDL to the pipeline pick up 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, this allows you to 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 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 adapter service is enabled as a Web service provider, and you want to receive Web service requests over HTTP, create a TCPIPSERVICE resource.

When you have completed these steps, the adapter service has been successfully deployed into CICS.

Updating an existing adapter service

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

- If you have deployed an adapter service using WebSphere Developer for System z version 6:
  1. Use the CEMT transaction to refresh the adapter in the appropriate CICS region. For example:

     ```
     CEMT SET PROGRAM (name) NEWCOPY
     ```

     Where *name* is the NAVIGATOR or COMMAND program id.
  2. Use the **NEWCOPY** command for any server adapters that were modified.
- If you have deployed an adapter service using WebSphere Developer for System z version 7:
  1. Disable the PROCESSTYPE resource for your adapter service using the command **CEMT SET PROCESSTYPE**. The name of the resource is the same as the request name of the flow that you modeled.
  2. If you are changing the associated resources, for example using a different transaction or program, delete the resources from the CSD.
  3. Regenerate the adapter service and run through the deployment steps as outlined in “Deploying a new adapter service” on page 63. Ensure that the **MODE** parameter on the properties file update job is set to `MODE=OVERWRITE`.
  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. Check that the PROCESSTYPE resource is enabled. If it's still disabled, enable it using the **CEMT SET PROCESSTYPE** command.

The properties file

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

When you deploy a new adapter service, the properties file is updated to include all of the information necessary to perform request processing for that adapter service. Each adapter service record begins with a type R record that describes the request. It can include information such as:

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

The type R record is followed by definitions for the server adapters that are used in the adapter service. The only server adapter that is not required in the properties file is the Web services server adapter.

The properties file is read by the DPL stub program, the navigation manager and server adapters during the processing of a request message.

To avoid problems at run time, it is important to ensure that you update the properties file correctly. By default, the properties file is updated in safe mode during the deployment process. This ensures that you do not overwrite any existing adapter service records if there is a naming conflict. If you want to update an existing adapter service record in the properties file, you should follow the deployment steps and update the properties file by setting the **MODE** parameter to overwrite the definitions.

To view the contents of the properties file, use the supplied sample job DFHMAMPD to dump it.

Chapter 6. Invoking an adapter service

Invoking an adapter service occurs when a service requester passes a *request message* to the appropriate stub program in CICS Service Flow Runtime using one of the supported interfaces.

The request message is comprised of one or more message headers and application data. It contains the parameters and information necessary to initiate runtime processing and to invoke an adapter service 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 adapter service, and the types of server adapter that comprise the service.

The service requester can use different interfaces to invoke an adapter service, but can only invoke one adapter service 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 Distributed Programming Link (DPL), you can choose to pass the request message to the appropriate stub 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.

If you want to use a channel and containers to invoke an adapter service, apply APAR PK32131 to enable support for this function.

The service requester

A service requester is the application that is looking for and invoking or initiating an interaction with a service.

The requester role can be played by a browser driven by a person, or a program without a user interface, e.g. a Web service. 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 is looking for and invoking an interaction with the adapter service 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 an WebSphere MQ-enabled service requester and CICS. The request message is passed in a WebSphere MQ message to a WebSphere MQ message queue.
Non-WebSphere MQ applications	A CICS-supplied interface, such as **EXEC CICS LINK**, EXCI or ECI. The IBM CICS Transaction Gateway (CTG) product.

© Copyright IBM Corp. 2001, 2010

Examples of service requesters could be any of the following applications:

- WebSphere MQ Integrator
- WebSphere MQ Workflow
- Web service
- Any local or remote application that is capable of initiating 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, including passthrough processing if used, 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 message for passthrough processing. See “XML request and response processing for passthrough” on page 188.
- Performing data conversion if required. See “Data conversion” on page 74 for information on how to perform data conversion.

If you want to implement a passthrough deployment type to process requests from the service requester, you must program the service requester to handle the passthrough response, one screen at a time.

For passthrough processing, the service requester is responsible for interpreting the application data structure (ADS) and based on its interpretation, taking appropriate action in the language of the service requester.

Invoking an adapter service using a CICS-supplied interface

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

1. Select the interface that is appropriate for your service requester.
   a. If you want to use CTG to invoke the adapter service, use the ECI interface. However, if CTG is installed on z/OS, the EXCI interface is actually used.
   b. If you want to use DPL, use the **EXEC CICS LINK** command to link from the service requester to the DPL stub program with a request message. If you opt for DPL, you can use a channel and containers to pass the request message, rather than a COMMAREA.
2. Decide what data conversion 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 74.
3. Decide how you want to send the request message from the service requester to invoke the adapter service. The service requester can send the message in a COMMAREA, or if you are using DPL a set of containers.

- If you want to use a set of containers, see “Sending the request message in containers” on page 71 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 73 for the required format.

The service requester then waits for a response if required.

4. When the adapter service has completed its processing of the request message, the DPL stub 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 an adapter service using WebSphere MQ triggering

If the service requester is WebSphere MQ-enabled, you are required to use the WebSphere MQ-CICS Bridge to pass the request message to the DPL stub program DFHMADPL.

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

To start the CICS bridge monitor program using WebSphere MQ triggering, define the WebSphere 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.

The following steps describe how the runtime is invoked in asynchronous mode. See Figure 24 on page 70 for an illustration of this process.

1. The service requester sends a request message to the request queue. This causes WebSphere MQ to send a trigger message to the specified initiation queue.
2. The WebSphere MQ trigger monitor program starts the CICS bridge monitor task, that is part of the WebSphere MQ-CICS 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, that is part of the WebSphere MQ-CICS bridge.
4. The CICS bridge link task links to the DPL stub program, passing the request message in a COMMAREA, and waits for control to be returned. The request message that is passed to the stub program does not include the WebSphere MQ header data. The DPL stub program defines and runs a BTS process, passing the request message and waits for the adapter service processing to complete.
5. If a reply message is required, the DPL stub 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 WebSphere MQ-CICS bridge header (MQCIH) is not present in the request message, it will not appear in the reply message, except in the case of an error.

- In asynchronous processing, the WebSphere MQ-CICS bridge header (MQCIH) structure is always included in the reply message.

6. The CICS bridge link task responds to the service requester using an **MQ PUT** command if the MQMD ReplyToQ and ReplyToQMgr are loaded.

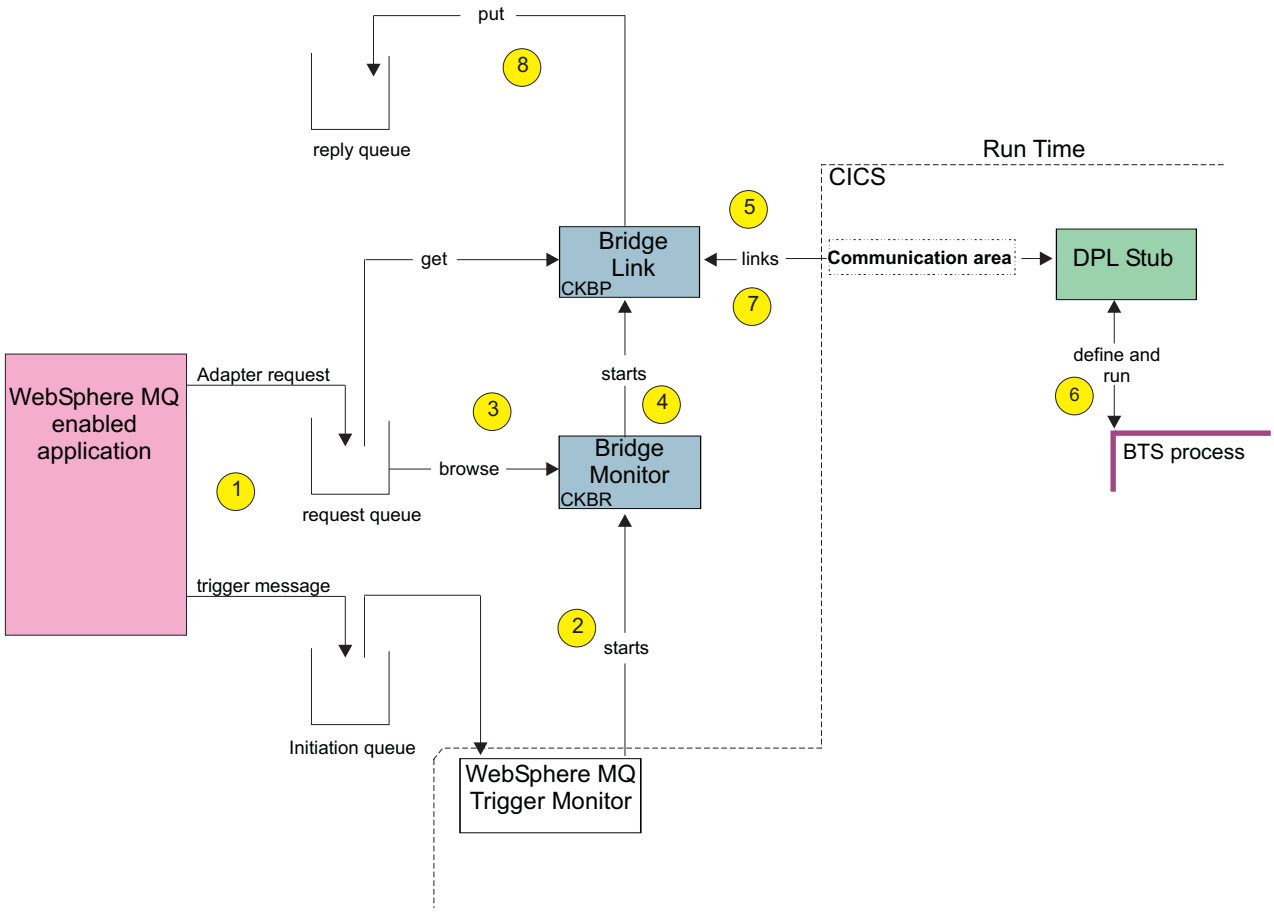


Figure 24. Invoking the run time via WebSphere MQ-CICS Bridge

Invoking an adapter service from a Web service

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

The provider PIPELINE resource must be enabled to process the Web service request and response messages. The WEBSERVICE and URIMAP resources for your Web service provider application must also be enabled. If the request is using HTTP, you should also have a TCPIPSERVICE resource that is enabled.

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 Modeler.
   - By default, the SOAP body includes the request name of the adapter service 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 is placed in the DFHWS-DATA container by default. If you select to expose the whole message header in Service Flow Modeler, the DFHMAC-ALLPARMS container is used instead.

2. The application handler in the pipeline passes the appropriate container to the DFHMADPL program.
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 adapter service using the values in the containers.
   - If the request message is successfully processed and a response is required, the DPL stub program 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 DPL stub program creates a SOAP fault message and this 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 need to remodel your flows or redeploy adapter services to use channels and containers.

You must have APAR PK32131 applied to pass the request message to CICS SFR using channels and containers.

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 adapter service that you want to invoke in the DFHMAC-REQUESTV1 container, and the application data in the DFHMAC-USERDATA container. The request name should match the PROCESSTYPE resource that is defined in CICS SFR for the adapter service. 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 DPL stub 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 adapter service as a Web service provider application. For details on how to do this, see “Invoking an adapter service from a Web service” on page 70.
   - 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.
   - If your request message requires passthrough processing, use the DFHMAC-PASSTHRU container. This container should contain the DFHMAH and DFHMAH2 headers and should be followed by any screen vectors.

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 adapter service.
   a. If you want to process the request message using a different process type, change the request name, or change the process name, use the DFHMAC-SYSPARMV1 container.
      - Specify a different process type to override the PROCESSTYPE resource that is defined in CICS for the adapter service. By default the request name that is passed in DFHMAC-REQUESTV1 matches the PROCESSTYPE resource of the adapter service.
      - 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 should be a unique identifier. By default, a unique identifier is generated by CICS when the BTS process is created by the stub program.

      This overrides the values associated with the adapter service when the request message is received.
   b. If you want 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 DFHMADPL stub program. For example, you could specify:

```
* 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)
```

   In the case of passthrough processing, DFHMADPL invokes DFHMADPP to perform request processing.

   When the request message has been processed, the response is placed in the appropriate container that holds the application data. For example, if the service requester passed the application data in DFHMAC-USERDATA, the response is placed in this container for the service requester to retrieve. The message header is also updated.

   If an error occurs, the DHFMAC-ERROR container is passed 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 or DFHMAC-PASSTHRU 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:

```
* 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 should 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 what data structure or application data format comes next in the request message.

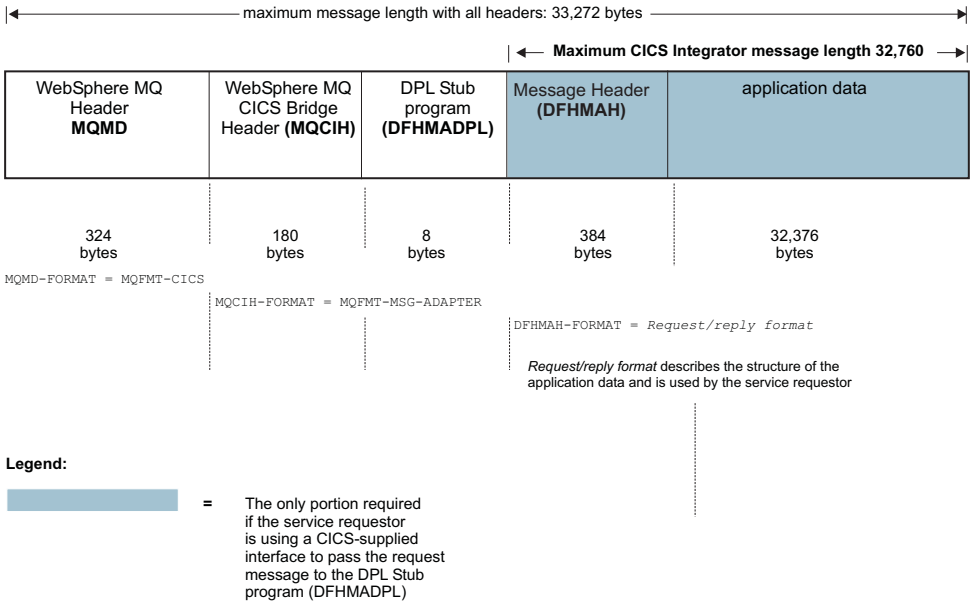
The headers that should be included in the request message depend on the interface that the service requester is using to invoke the adapter service. You also need to provide additional headers if you want to invoke a CICS application using passthrough processing and the Link3270 bridge.

- If you want the service requester to invoke an adapter service 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 utilize the CICS Transaction Gateway (CTG) interface, the request message should contain the 101-byte CTG header, followed by the DFHMAH message header and then the application data.
- If the service requester is an WMQ-enabled application, and you want to invoke an adapter service, the request message should include a WMQ header (MQMD), the WebSphere MQ CICS Bridge header (MQCIH), the DFHMADPL stub program header, and the message header DFHMAH.

  The MQCIH header is only required when you are implementing password authentication. However, it should always be included because this enables 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 requestor. This process occurs regardless of whether the structure was sent in the request message.

  The following figure illustrates the structure of this request message.

Request message structure for non-passthrough requests

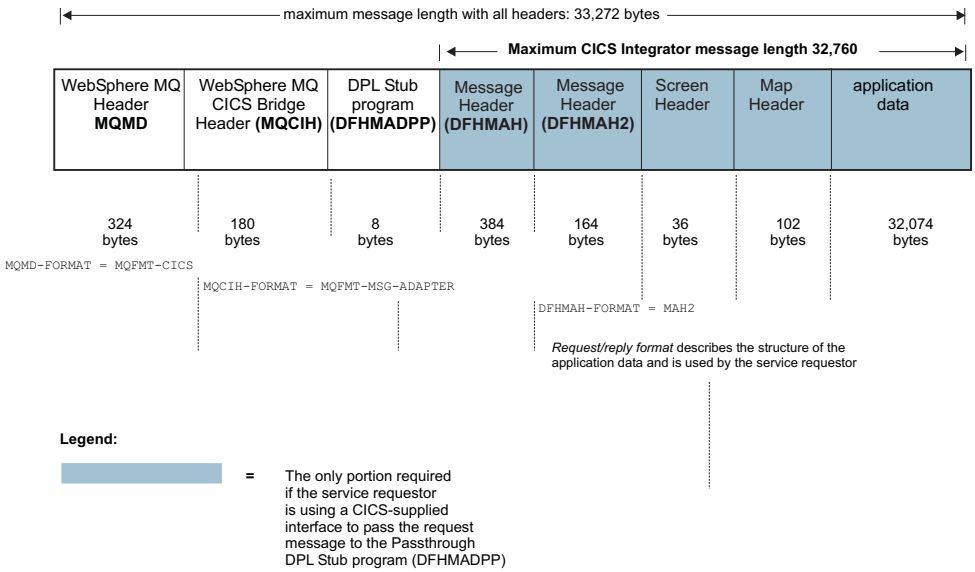


- If you want the service requester to use a CICS-supplied interface or the CTG to invoke a CICS application using passthrough processing, include the DFHMAH and DFHMAH2 headers followed by the application data.

- If the service requester is an WMQ-enabled application, and you want to invoke a CICS application using passthrough processing and the Link3270 bridge mechanism, the request message should include a WMQ header (MQMD), the WebSphere MQ CICS Bridge header (MQCIH), the DFHMADPP stub program header, the message header DFHMAH, and the message header DFHMAH2. In addition, the screen header CIA-SCREEN-HEADER and the map header CIA-MAP-HEADER must be included after the DHMAH2 header.

  The following figure illustrates the structure of this request message.

Request message structure for passthrough requests



Related reference:

"Request message headers" on page 80
The structure and fields of the request message headers are described in this section. These headers, along with the application data, comprise the request message that is sent by the service requester in a COMMAREA.

Data conversion

Data conversion refers to the process of changing data from one form of representation to another. A system might need to 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 message or reply messages. If data conversion is required, it is the responsibility of the service requestor to implement it. The following sections explain how to implement data conversion when using the supported interfaces from the service requestor to the CICS Service Flow Runtime environment.

Data conversion using the WebSphere MQ interface

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

If the service requester is a WebSphere MQ-enabled application, it is required to use the WebSphere MQ-CICS Bridge when invoking the CICS Service Flow Runtime.

When data conversion is required, it becomes 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 *WebSphere MQ Application Programming Guide*.

If the service requester is required to convert application data between different machine encoding and CCSIDs, the service requester must conform to the WebSphere 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 *WebSphere MQ Application Programming Reference*.

Data conversion using a CICS-supplied interface

If you are using a CICS-supplied interface to invoke an adapter service, it might be necessary to perform data conversion in the CICS Service Flow Runtime routing region utilizing a customized version of the standard CICS conversion table, DFHCNV.

A data conversion table could be needed 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 to be changed from ASCII format at the workstation to EBCDIC format in CICS.

ECI applications use the facilities of 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 nonstandard, user-handled conversion is not specified.

If DFHCCNV finds a conversion template in the DFHCNV table that matches the runtime initial program name (the DPL Stub program name of DFHMADPL or DPL Passthrough Stub program name of DFHMADPP), 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 that should be converted, and the conversion method to be applied to each field.

The conversion template must specify:

- A table entry for the initial DPL stub program name, DFHMADPL or DFHMADPP.
- Entries for each field offset, type of conversion and field length found in the message header structures, DFHMAH and DFHMAH2.
- All binary fields must specify DATATYP=NUMERIC if ASCII to EBCDIC conversion is required (actually little-endian to big-endian conversion). Similar conversion entries (DATATYP=NUMERIC) with correct field offsets must be supplied if binary data is used in the client application data portion of the request and reply messages.
-

See “Conversion templates” on page 316 for samples of the runtime conversion templates for DFHMADPL and DFHMADPP.

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.

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 could 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.

For detailed information about the DFHCCNV conversion program, conversion templates, the DFHCNV conversion table, and the syntax of DFHCNV macros, see the following publications:

- *CICS Family: Communicating from CICS on System/390® SC34-6031-05*
- IBM redbook: *CICS Transaction Gateway V5: The WebSphere Connector for CICS SG24-6133-01*.

There are other methods that can be used to perform Unicode or ASCII to EBCDIC data conversion. For example, conversion within a Java application. See the following publications for further information regarding these methods:

- *Java applications in CICS*
- IBM Redbook: *Java Connectors for CICS SG24-6401-01*

For information on how to install and configure DFHCNV, see “Setting up data conversion” on page 29.

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, it is recommended that a split-client design is employed 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

The structure of the request message containers are described in this section. Use these containers to send the request message header and the application data, along with optional parameter values to override defaults in the runtime environment.

Channel and container support is only available when you apply APAR PK32131.

Container DFHMAC-ALLPARMS

DFHMAC-ALLPARMS is used by the service requester to pass the contents of the DFHMAH header to the stub program.

This container is provided 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 would use to pass a request message in a COMMAREA. The container is mapped by the copybook DFHMAHV, which is located 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 is in XML, then 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, 16MB in length.

Container DFHMAC-ERROR

DFHMAC-ERROR is used by CICS Service Flow Runtime 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 located in the SCIZMAC library.

The following table lists all of the fields that are included in the container, although typically not all of the fields would be set for an error. A field's type can be:

- X - the field contains characters.
- FB - the field is a fullword binary.

Field	Length (bytes)	Type	Description
DFHMAHE-RETURNCODE	04	FB	The return code from CICS Service Flow Runtime that describes the outcome of adapter request processing. The value is one of the following: • 9 = Error • 99 = Multiple errors • 999 = Abend
DFHMAHE-COMPCODE	04	FB	The completion status of the BTS process instance. This describes the outcome of the BTS process that implements an instance of the CICS Service Flow Runtime.
DFHMAHE-MODE	04	FB	The processing state of the BTS process instance. This 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.
DFHMAHE-SUSPSTATUS	04	FB	The suspend status of the BTS process.
DFHMAHE-ABENDCODE	04	X	The ABEND code. This field is only set when the return code is 999.
DFHMAHE-MESSAGE	12	X	The error message returned byCICS Service Flow Runtime.
DFHMAHE-FAILED-PROCNAME	36	X	The name of the failed BTS process.
DFHMAHE-FAILED-PROCTYPE	08	X	The type of the BTS process that failed. The process type usually matches the request name of the adapter service.

Field	Length (bytes)	Type	Description
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	The return code that is issued by the Link3270 bridge mechanism to the CICS Service Flow Runtime.
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

DFHMAC-LNK3270V1 is used by the service requester 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 located 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-PASSTHRU

DFHMAC-PASSTHRU is used by the service requester to access the passthrough processing in the runtime environment.

This container holds the DFHMAH and DFHMAH2 headers that provide the passthrough processing with the information necessary to carry out the request. It also contains any application data that is required. The maximum length of the container, including the headers, is 32,500 bytes.

The container is mapped by copybooks DFHMAHV and DFHMAH2V, which are located in the SCIZMAC library.

Container DFHMAC-REQUESTV1

DFHMAC-REQUESTV1 is used by the service requester to pass the request name of the adapter service to the DPL stub program DFHMADPL.

The request name is 8 characters in length and, by default, is the same as the PROCESSTYPE resource for the service adapter. When this container is passed to the stub program, the default values for the header are used in the request

processing. These default values can be overridden by passing in additional containers. The container is mapped by the DFHMAHRV copybook, which is located in the SCIZMAC library.

Do not include any application data in this container.

Container DFHMAC-SYSPARMV1

DFHMAC-SYSPARMV1 is used by the service requester 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 located in the SCIZMAC library. If a parameter value is non-blank, then 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 adapter service

Container DFHMAC-USERDATA

DFHMAC-USERDATA is used by the service requester to pass application data to the stub program DFHMADPL.

This is an optional container that can be sent by the service requester along with the DFHMAC-REQUESTV1 or DFHMAC-ALLPARMS containers. When the adapter service 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, 16MB.

Container DFHWS-DATA

DFHWS-DATA is used to pass the request name and application data to the stub program DFHMADPL when a Web service request is received and processed in a provider mode pipeline.

The request name is 8 characters in length, and by default is the same as the PROCESSTYPE resource for the service adapter. When this container is passed by the pipeline application handler to the stub program, the default values for the header are used in request processing. These default values can be overridden by passing in additional containers.

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

Request message headers

The structure and fields of the request message headers are described in this section. These headers, along with the application data, comprise the request message that is sent by the service requester 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

Related tasks:

"Sending the request message in a COMMAREA" on page 73
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 what data structure or application data format comes next in the request message.

DFHMAH header structure

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

The following table highlights field information within 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 4. 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, MAH2
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, +3, +9
64	08	x	DFHMAH-PROCESSTYPE	Y	
72	36	x	DFHMAH-PROCESSNAME	N	

Table 4. DFHMAH message header fields (continued)

Disp.	Length	Type	Field	Req.	Values
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	zero, +1
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	

Related reference:

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

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 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 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 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. This is the format name of the application data that follows the DFHMAH structure or if using the Link3270 Passthrough, the STRUCID of the CICS Service Flow Runtime header that follows (MAH2). 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 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 adapter request processing. The Compcode, Mode, Suspstatus, Abendcode and Message fields can contain additional information. The value is one of the following:

- 0 = Normal
- 9 = Error
- 99 = Multiple errors
- 999 = Abend

Depending on the processing mode (asynchronous, synchronous, or synchronous rollback), a value other than zero in this field can indicate an incomplete BTS process. This process will remain in an incomplete status until some action is taken. See the description of field DFHMAH-COMPCODE.

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

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

DFHMAH-MODE
The processing state of the BTS process instance. This 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. The initial value of this field is zero. See the *CICS Business Transaction Services* manual for a description of the **CHECK ACQPROCESS** command.

DFHMAH-SUSPSTATUS
The suspend status of the BTS process. This is an output field. The initial value of this field is zero. See the *CICS Business Transaction Services* manual for a description of the **CHECK ACQPROCESS** command.

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

DFHMAH-MESSAGE
The error message returned from CICS Service Flow Runtime. The value returned in this field is dependent on the DFHMAH-RETURNCODE field. This is an output field and the initial value is blanks. See “Error messages” on page 220 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 is the initial value of the field. A value of 0 indicates standard sequence flow processing of an adapter service that was modeled and generated from Service Flow Modeler and deployed to the CICS Service Flow Runtime environment, where a BTS process instance is created to execute an inbound request from a service requestor.

- 2 = compensate

  A value of 2 indicates that should a BTS process instance fail, the process is cancelled and a compensating flow runs instead.

- 3 = passthrough processing (applies only to Link3270 server adapters)

  A value of 3 indicates a passthrough processing mode, where no modeled, generated or deployed Adapter service are executed. Instead, the CICS Service Flow Runtime initiates and processes CICS target applications based on the information that was contained in the request message. This processing mode also creates a BTS process instance to execute the inbound passthrough request. This processing mode always requires the header structures DFHMAH2, CIA-SCREEN-HEADER and CIA-MAP-HEADER

- 9 = cancel

  A value of 9 indicates that should a BTS process instance fail, the process is cancelled.

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

DFHMAH-PROCESSTYPE

This field categorizes and defines the type of the new process instance. It is controlled by the service requestor. The value in this field must be defined to CICS. The CICS definition is done through resource definition online (RDO) or through CICSPlex® SM Business Application Services (BAS) and assigns resources (repository, audit log). It is used on the **BTS DEFINE PROCESS** command in the DPL Stub program (DFHMADPL or DFHMADPP).

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.

This field does not need to be specified if the DFHMAH-UOWCONTROL field has a value of 9. Due to the relationship between 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. It is used on the **BTS DEFINE PROCESS** command in the DPL Stub program (DFHMADPL or DFHMADPP). This field ensures that each BTS process has a unique name. See the *CICS Business Transaction Services* manual for a description. This is an input or output field. The initial value of this field is blanks.

If no value is provided, the CICS Service Flow Runtime will generate a unique identifier and return it to the service requestor. When generated, the user id for which the Stub program is running, Eibtaskn, and the AbsTime values are concatenated in that order and are used as the process name.

This field does not need to be specified if the DFHMAH-UOWCONTROL field has a value of 9.

DFHMAH-REQUESTNAME

This field indicates the name of the request to process. The value specified is used to read the CICS Service Flow Runtime Properties file to determine processing flow and parameters. The value must correspond to the name on a TYPE=R Property file record. The name can specify either a new request name or a request name of a compensation request. A compensation request name means that the flow will be used to compensate for a previously failed CICS Service Flow Runtime process. This is an input field. See “How compensation processing works” on page 193 for a description of compensation processing.

The initial value of this field is blanks. This field does not need to be specified if the DFHMAH-UOWCONTROL field has a value of 9.

Note: The request name (**DFHMAH-REQUESTNAME**) must be passed in as part of the request message in the message header structure (**DFHMAH**). In CICS Service Flow Runtime passthrough processing, the Properties file is not used. However, the request name still must be passed in as part of the request message as it is used as the activity name on the **BTS DEFINE ACTIVITY** command when defining the Passthrough Manager (DFHMALPT) activity.

DFHMAH-DATALENGTH

This field indicates the length of the inbound request or outbound reply data following this header structure data inclusive of any CICS Service Flow Runtime passthrough header structures, if present, and any application request/reply data. This length does not include the lengths of any WebSphere MQ header structures and the length of this DFHMAH header structure.

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, XML declaration data, etc., if present.

For passthrough and non-passthrough XML request messages, this field value and definition are different depending on the type of request being processed by the CICS Service Flow Runtime as follows:

- For non-passthrough XML requests, this field should indicate the length of the application request data in fixed format (i.e., the length of the COBOL fixed length group item copybook generated and deployed by the WebSphere Developer for System z tool that maps this application request data used in the deployed Adapter service).
- For XML and non-XML passthrough requests, this field should indicate the total of the fixed format length of the passthrough header structure, DFHMAH2 (+164 bytes), plus the fixed format length of the CIA-SCREEN-HEADER structure (+36 bytes) plus the fixed format length of the CIA-MAP-HEADER structure (+102 bytes) plus the fixed format length of any application data structure (ADS), 3270 datastream or unformatted application request data if present.

  This field should NOT indicate the length of the XML formatted request application data on input. The field value may be larger than the actual length of the request data passed but should not be smaller than required or data truncation may occur causing unexpected results.

This field is input on requests and output on replies. For inbound non-passthrough request messages, this field determines the length of the input data container that holds the request application data. The initial value

of this field is zero. The request data length is determined by the service requestor. The reply data length is set by the CICS Service Flow Runtime.

DFHMAH-FAILED-PROCNAME
Failed process name. This field indicates the name of the failed process. This 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. It must be specified 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 is an input or output field. It is returned by the CICS Service Flow Runtime when *RETURNCODE* is not zero. It must be specified 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 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. It can be used by the application 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 requestor upon failure.

DFHMAH-REPLYTOQ
Name of the reply queue. This is the name of the message queue to which the CICS Service Flow Runtime should send reply messages. This is an input field. See the*WebSphere MQ Application Programming Reference* for a description. This field must be specified if the processing request mode indicates *asynch*, and the WebSphere MQ CICS bridge is being used to communicate between the CICS Service Flow Runtime and the service requestor. In this case, ReplyToQ should not be specified in the WebSphere MQ MQMD structure.

This field can be loaded if you are implementing early reply processing (MsgType = +2) with the queue name. The name will be used for any early reply issued. This field has no effect on MsgType +8 (Datagrams). MsgType + 8 requires the queue name be specified on the properties file.

DFHMAH-REPLYTOQMGR
This is the name of the queue manager to which the CICS Service Flow Runtime should send reply messages. This is an input field. *Replytoq* is the local name of a queue that is defined on this queue manager. See the *WebSphere MQ Application Programming Reference* for a description. This field must be specified if the processing request mode indicates *asynch*, and the WebSphere MQ-CICS bridge is being used to communicate between the CICS Service Flow Runtime and the service requestor. In this case, ReplyToQMgr should not be specified in the WebSphere MQ MQMD structure.

This field can be loaded if the customer is implementing early reply processing (MsgType = +2) with the queue manager name. The name will be used for any early reply issued. This field has no effect on MsgType +8 (Datagrams). MsgType + 8 requires the queue name be specified on the properties file.

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 (update) processing, CICS Service Flow Runtime handles the reply message itself, rather than allowing the bridge to return the reply message. The CICS Service Flow Runtime is not passed the WMQ headers (containing the MSGID) by the bridge. 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 **PUT** commands issued from CICS Service Flow Runtime server adapters and on any reply messages to the service requestor. It 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. CICS Service Flow Runtime is not passed the MQ headers (containing the CORRELID) by the bridge. Therefore, when you want your adapters to know what correlid the message was put with, you must explicitly supply a CORRELID, rather than allowing it to be generated by the queue manager. It is an input field. Note that 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 is an input or output field. It is returned when the value of field DFHMAH-RETURNCODE is not zero. It can be used by the application 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 requestor upon failure.

DFHMAH-FAILED-NODE

This field indicates the name of the active node when an error occurred causing process failure. This 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 It can be used by the application 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 requestor upon failure. See the information on the Service Flow Modeler in the WebSphere Developer for System z information center for detailed information on defining nodes.

DFHMAH-LINKTYPE

This field indicates the behavior of the interface used to initiate the CICS Service Flow Runtime. The initial value of this field is zero. Acceptable values are as follows:

- zero = asynchronous interface (MQSeries-CICS bridge)
- +1 = synchronous interface (DPL, EXCI, ECI i.e., CICS or CICS Transaction Gateway)

The value specified in this field overrides the corresponding value on the properties file for the request. So, if the **request type** property is set to asynchronous (Request Type = 0) on the Properties file, but DFHMAH-LINKTYPE = 1 (synchronous) on the request message, then the request processing associated with this process, will be synchronous.

DFHMAH-MORE-DATA-IND

This field indicates if additional response data is available but could not be delivered. It is returned by the CICS Service Flow Runtime when the response data is larger than 32,000 bytes. This 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 mechanism 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 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 is an input and output field. It is returned when request processing is complete. It may be passed in on 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 *CICS External Interface Guide Version 2 Release 2* or higher for more information on Link3270 bridge facilities.

DFHMAH-RESERVED2

Reserved.

Related reference:

"DFHMAH header structure" on page 80
The request message header DFHMAH describes the structure of the application data. It is mapped by copybook DFHMAHV, which is located in the library SCIZMAC.

DFHMAH2 header structure

The request message header DFHMAH2 describes the structure of the application data for a passthrough request message. It is mapped by copybook DFHMAH2V, which is located in the library SCIZMAC.

When the DFHMAH-UOWCONTROL field in DFHMAH has a value of `3`, the DFHMAH2 header structure is required. The DFHMAH-FORMAT field in DFHMAH must also have a value of `MAH2` to indicate the existence of the DFHMAH2 header structure in the inbound passthrough request.

Note: The associated headers CIA-SCREEN-HEADER and CIA-MAP-HEADER are mapped by copybooks DFHMALSH and DFHMALMH respectively. These copybooks are also located in the library SCIZMAC.

The following table highlights field information within the DFHMAH2 message header structure.

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 5. DFHMAH2 passthrough message header fields

Disp.	Length	Type	Field	Req	Values
0	04	x	DFHMAH2-STRUCID	Y	MAH2
4	04	FB	DFHMAH2-VERSION	Y	+1
8	04	FB	DFHMAH2-STRUCLENGTH	Y	+164
12	08	x	DFHMAH2-RESERVED	N	
20	08	x	DFHMAH2-FORMAT	N	
28	04	FB	DFHMAH2-DATALENGTH	Y	
32	04	x	DFHMAH2-TRANSID	N	
36	04	FB	DFHMAH2-RECEIVE-TYPE	Y	zero, +1, +2
40	04	x	DFHMAH2-NEXT-TRANSID	N	
44	04	FB	DFHMAH2-USE-FKEEPTIME-IND	N	
48	04	FB	DFHMAH2-FACILITYKEEPTIME	N	
52	04	x	DFHMAH2-FACILITYLIKE	N	
56	04	FB	DFHMAH2-GETWAITINTERVAL	N	
60	04	FB	DFHMAH2-VECTOR-LOGGING	N	zero, +1, +2
64	04	FB	DFHMAH2-DEALLOCATE-IND	N	zero, +1, +2, +3
68	04	FB	DFHMAH2-SEND-AID-FIRST	N	zero, +1
72	01	x	DFHMAH2-INITIAL-AID-BYTE	N	
73	39	x	DFHMAH2-CLIENTIP-ADDRESS	N	
112	04	FB	DFHMAH2-RESPTIME	N	
116	04	FB	DFHMAH2-APPLRESPTIME	N	
120	08	x	DFHMAH2-XML-PROGRAMID	N	
128	36	x	DFHMAH2-RESERVED2	N	

Related reference:

"DFHMAH2 field definitions"
The following definitions describe the content of each field in the CICS Service Flow Runtime passthrough message header (DFHMAH2).

"CIA-SCREEN-HEADER header structure" on page 92
When the runtime processing is running in passthrough mode, the CIA-SCREEN-HEADER structure is required. It must immediately follow the DFHMAH2 header structure. To view the passthrough screen header, see copybook DFHMALSH in the SCIZMAC library.

"CIA-MAP-HEADER header structure" on page 94
When the server runtime processing is running in passthrough mode, the CIA-MAP-HEADER structure is required. It must immediately follow the CIA-SCREEN-HEADER structure. To view the passthrough map header, see copybook DFHMALMH in the SCIZMAC library.

DFHMAH2 field definitions

The following definitions describe the content of each field in the CICS Service Flow Runtime passthrough message header (DFHMAH2).

DFHMAH2-STRUCID

Structure Identifier. The value must be MAH2. This is always an input field. The initial value of this field is MAH2.

DFHMAH2-VERSION

Structure version number. The value must be +1 for CICS Service Flow Runtime processing. The field can have the following values:

- +1 = CICS Service Flow Runtime structure version

This is always an input field. The initial value of this field for CICS Service Flow Runtime = +1.

DFHMAH2-STRUCLENGTH

Length of DFHMAH2 structure. The value must be +164. This is always an input field. The initial value of this field is +164.

DFHMAH2-RESERVED

Reserved.

DFHMAH2-FORMAT

CICS Service Flow Runtime request / reply name format name. This is the format name of the application message data that follows the DFHMAH2 structure. This is an input or output field. The initial value of this field is spaces. This is an input or output field. The initial value of this field is spaces.

DFHMAH2-DATALENGTH

This field indicates the length of the inbound application request or outbound application reply data following this header structure data inclusive of the CICS Service Flow Runtime passthrough header structures CIA-SCREEN-HEADER and CIA-MAP-HEADER.

This length does not include the lengths of any WebSphere MQ header structures, the length of the DFHMAH header structure and the length of this DFHMAH2 header structure. 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, and XML declaration data, if present.

For CICS Service Flow Runtime XML and non-XML passthrough requests, this field should indicate the total of the fixed format length of the CIA-SCREEN-HEADER structure (+36 bytes) plus the fixed format length of the CIA-MAP-HEADER structure (+102 bytes) plus the fixed format length of any application data structure (ADS), 3270 data-stream or unformatted application request data if present. This field **should NOT indicate** the length of the XML formatted request application data.

This field **should NOT indicate** the length of the XML formatted request application data on input. The field value may be larger than the actual length of the request data passed but should not be smaller than required or data truncation may occur causing unexpected results.

Note: For inbound XML requests that include tag data that is to be placed in an application data structure (ADS) for a subsequent RECEIVE MAP, the length to be used for the application data length used in the calculation above is the length of the ADS not the length of the inbound XML formatted application request data. The length of the ADS can be found in the field CIAMH-MAP-DATA-LENGTH within the CIA-MAP-HEADER structure when a SEND MAP vector is returned.

For information on how the CICS Service Flow Runtime processes an XML message header, see “XML request and response processing” on page 182

This field is input on requests and output on replies. For inbound request messages, this field determines the length of the input data container that

holds the request application data. The initial value of this field is zero. The request data length is determined by the service requestor. The reply data length is set by the CICS Service Flow Runtime.

DFHMAH2-TRANSID
This field indicates the target application transaction ID to run. It is an optional field but must be provided on the initial application request. This is an input field.

DFHMAH2-RECEIVE-TYPE
This field indicates the inbound vector type that the runtime must build and use to satisfy the previous outbound vector type sent from the target CICS application. The value is one of the following:

- 0 = No RECEIVE
- +1 = RECEIVE MAP vector
- +2 = RECEIVE vector

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

A RECEIVE vector (+2) should be used when initiating user transactions with unformatted input data via the Link3270 bridge mechanism. See "CIA-MAP-HEADER field definitions" on page 95 for further information.

DFHMAH2-NEXT-TRANSID
This field indicates the next target CICS application transaction ID to run as indicated by the target CICS application. This is an output field. The initial value of this field is blanks.

DFHMAH2-USE-FKEEPTIME-IND
This field indicates if the value in DFHMAH2-FACILITYKEEPTIME should be used as the bridge facility maximum *keeptime*. The value is one of the following:

- Zero = Do not use DFHMAH2-FACILITYKEEPTIME
- +1 = Use

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

For information on FACILITYKEEPTIME, see the *CICS External Interfaces Guide*.

DFHMAH2-FACILITYKEEPTIME
This field defines the maximum length of time the bridge facility will be kept if inactive (i.e., keeptime).

Bridge facilities are deleted automatically if they are inactive for the keeptime interval. This field value is used to load the bridge header (BRIH) field BRIH-FACILITYKEEPTIME. If the bridge facility is not explicitly deleted, it is scheduled for deletion automatically by CICS if it is unused for the time specified in the BRIH-FACILITYKEEPTIME field, or in the **BRMAXKEEPTIME** system initialization parameter. The smaller interval is used. If a value is not provided for this field, a default value of 300 seconds is used unless overridden by **BRMAXKEEPTIME**.

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

DFHMAH2-FACILITYLIKE
This field defines the FACILITYLIKE value to be used on a bridge facility allocate request. The value is the name of a real terminal resource definition that is used as a template for some of the properties of the bridge facility. With the Link3270 bridge mechanism, you do not provide a TERMINAL resource definition for the bridge facility, but you can control the

terminal properties used by providing a 3270 TERMINAL resource definition to be used as a template. This TERMINAL definition is known as the FACILITYLIKE. This field value is used to load the bridge header (BRIH) field BRIH-FACILITYLIKE on the allocate facility request. If the facility is reused, this value is ignored on subsequent calls to the bridge mechanism.

This is an input field. The initial value of this field is spaces. For information on FACILITYLIKE, see *CICS External Interfaces Guide*

DFHMAH2-GETWAITINTERVAL
This field defines the maximum wait interval for message input (in milliseconds). This field value is used to load the bridge header (BRIH) field BRIH-GETWAITINTERVAL on bridge requests. The actual value used is the smaller of the BRIH-GETWAITINTERVAL and the RTIMEOUT value for the target CICS transaction.

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

DFHMAH2-VECTOR-LOGGING
This field indicates whether vector logging is enabled and if so the type of logging to be performed. Valid values for this field are as follows:

- Zero = no vector logging
- +1 = full vector logging (vector headers (BRIH/BRIV) and vector data)

  Note: When using full vector logging (MP-BR-VECTOR-LOGGING = +1; see "Link3270 node properties" on page 280) the vector data that will display as contents in the vector dump is the accumulated map buffer or text data that is returned to the generated Link3270 navigator. The buffer contents in the vector dump **may not always contain the individual vector data** returned from the Link3270 bridge mechanism.
- +2 = trace logging (vector headers (BRIH/BRIV) only)

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

DFHMAH2-DEALLOCATE-IND
This field indicates when to deallocate a Link3270 facility and delete any associated session state data. Valid values for this field are as follows:

- Zero = don't deallocate
- +1 = deallocate
- +2 = deallocate if processing is successful
- +3 = deallocate if processing is unsuccessful

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

DFHMAH2-SEND-AID-FIRST
This field indicates if an initial AID byte is to be sent to satisfy a RECEIVE/RECEIVE MAP vector or to clear a pseudo-screen before processing additional inbound vectors. The AID byte to send is found in field DFHMAH2-INITIAL-AID-BYTE. Valid values for this field are as follows:

- Zero = do not send an initial AID byte
- +1 = send an initial AID

This is an input field.

DFHMAH2-INITIAL-AID-BYTE
This field indicates the initial AID byte (pseudo-function key) to be sent to satisfy a RECEIVE/RECEIVE MAP/CONVERSE vector or to clear a

pseudo-screen of data if DFHMAH-SEND-AID-FIRST indicates to do so. Valid values for this field can be found in the DFHAID CICS copybook.

This is an input field. The initial value of this field is spaces.

DFHMAH2-CLIENTIP-ADDRESS
This field defines the client IP. Use is optional. May be used to identify a workstation or DHCP hub which may then map to a local CICS owned printer. Many CICS applications map a physical terminal to a physical printer. For example, several terminals may all be mapped to the same printer in an office location. This field is intended to provide similar support for a Web interface to CICS applications. If the Java client can provide the client IP address then it may be possible to map the address to a CICS owned printer. This is an input field. The initial value of this field is spaces.

Note: This field is not currently implemented.

DFHMAH2-RESPTIME
This field indicates the CICS Service Flow Runtime response time in milliseconds. Application response time is included in this response time value.

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

DFHMAH2-APPLRESPTIME
This field indicates the CICS target application response time in milliseconds.

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

DFHMAH2-XML-PROGRAMID
This field indicates, if the request and reply are in XML format, the name of the XML parse program to call for application data request/reply portion of the CICS Service Flow Runtime message structure. If the elementary item, DFHMAH-XML-PROGRAM-TAG, is left blank, a call to program name = DFHMAH-XML-PROGRAM + "I" will be made to parse XML application request data during request processing and a call to program name = DFHMAH-XML-PROGRAM + "O" will be made to generate XML application reply data during reply processing. If the DFHMAH-XML-PROGRAM-TAG is not equal to spaces, the program called will be the same to parse XML application request data on input and to build XML application reply data on output. This is an input field. The initial value of this field is spaces. See “XML request and response processing” on page 182

Note: This field is not currently implemented.

DFHMAH2-RESERVED2
Reserved.

Related reference:

“DFHMAH2 header structure” on page 87
The request message header DFHMAH2 describes the structure of the application data for a passthrough request message. It is mapped by copybook DFHMAH2V, which is located in the library SCIZMAC.

CIA-SCREEN-HEADER header structure

When the runtime processing is running in passthrough mode, the CIA-SCREEN-HEADER structure is required. It must immediately follow the DFHMAH2 header structure. To view the passthrough screen header, see copybook DFHMALSH in the SCIZMAC library.

The following table highlights field information within the passthrough screen header structure (CIA-SCREEN-HEADER).

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 6. Passthrough screen header fields

Disp.	Length	Type	Field	Req	Values
0	05	x	CIASH-STRUCID	Y	CIASH
5	01	x	CIASH-ATTENTIONID	Y	DFHAID values
6	03	9	CIASH-MAP-COUNT	N	
09	27	x	CIASH-RESERVED	N	

Related reference:

"DFHMAH2 header structure" on page 87
The request message header DFHMAH2 describes the structure of the application data for a passthrough request message. It is mapped by copybook DFHMAH2V, which is located in the library SCIZMAC.

"CIA-SCREEN-HEADER field definitions"
The following definitions describe the content of each field in the CICS Service Flow Runtime passthrough Screen header (CIA-SCREEN-HEADER).

CIA-SCREEN-HEADER field definitions

The following definitions describe the content of each field in the CICS Service Flow Runtime passthrough Screen header (CIA-SCREEN-HEADER).

CIASH-STRUCID

Structure Identifier. The value must be CIASH. This is always an input field. The initial value of this field is CIASH.

CIASH-ATTENTIONID

This field indicates the AID byte (pseudo-function key) to be used to satisfy a RECEIVE / RECEIVE MAP / CONVERSE vector or to clear a pseudo-screen of data. Valid values for this field can be found in the DFHAID CICS provided copybook.

This is an input field. The initial value of this field is spaces.

For further information, see the *CICS Application Programming Guide* and the *CICS Application Programming Reference*.

For XML <screen attentionid=tag values, see Table 7.

For information on CICS Service Flow Runtime XML message formats, see "XML message formats for passthrough requests" on page 340

Table 7. XML tag value to CICS Constant

XML tag value	Constant	Meaning
enter	DFHENTER	ENTER key
clear	DFHCLEAR	CLEAR key
pen	DFHPEN	Selector pen or CURSOR SELECT key
opid	DFHOPID	OPERID or magnetic slot reader (MSR)

Table 7. XML tag value to CICS Constant (continued)

XML tag value	Constant	Meaning
msre	DFHMSRE	Standard MSR
trig	DFHTRIG	Trigger field
pa1-pa3	DFHPA1-DFHPA3	PA1-PA3 keys
pf1-pf24	DFHPF1-DFHPF24	PF1-PF24 keys

CIASH-MAP-COUNT
This field indicates the number of map headers contained in the remaining request/reply data structure. Currently CICS Service Flow Runtime supports only 1 map header and any associated unformatted or formatted ADS data on inbound passthrough requests to either start a new transaction or to satisfy a RECEIVE/RECEIVE MAP/CONVERSE vector.

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

CIASH-RESERVED
Reserved.

Related reference:

"CIA-SCREEN-HEADER header structure" on page 92
When the runtime processing is running in passthrough mode, the CIA-SCREEN-HEADER structure is required. It must immediately follow the DFHMAH2 header structure. To view the passthrough screen header, see copybook DFHMALSH in the SCIZMAC library.

CIA-MAP-HEADER header structure

When the server runtime processing is running in passthrough mode, the CIA-MAP-HEADER structure is required. It must immediately follow the CIA-SCREEN-HEADER structure. To view the passthrough map header, see copybook DFHMALMH in the SCIZMAC library.

The following table highlights field information within the passthrough map header structure CIA-MAP-HEADER.

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 8. Passthrough map header fields

Disp.	Length	Type	Field	Req	Values
0	05	x	CIAMH-STRUCID	Y	CIAMH
5	01	x	CIAMH-DATA-FORMAT-INDICATOR	Y	I, B, 3, T
6	05	9	CIAMH-MAP-DATA-LENGTH	N	
11	8	x	CIAMH-MAP-NAME	N	
19	08	9	CIAMH-MAPSET-NAME	N	
27	03	9	CIAMH-MAP-ROWS	N	
30	03	9	CIAMH-MAP-COLUMNS	N	
33	05	9	CIAMH-FIELD-COUNT	N	

Table 8. Passthrough map header fields (continued)

Disp.	Length	Type	Field	Req	Values
38	01	9	CIAMH-EXT-ATTRIBUTE-COUNT	N	
39	07	x	CIAMH-EXT-ATTRIBUTES	N	
46	32	x	CIAMH-CURSPOS-FIELDNAME	N	
78	24	x	CIAMH-RESERVED	N	

Related reference:

"DFHMAH2 header structure" on page 87
The request message header DFHMAH2 describes the structure of the application data for a passthrough request message. It is mapped by copybook DFHMAH2V, which is located in the library SCIZMAC.

"CIA-MAP-HEADER field definitions"
The following definitions describe the content of each field in the CICS Service Flow Runtime passthrough map header (CIA-MAP-HEADER).

CIA-MAP-HEADER field definitions

The following definitions describe the content of each field in the CICS Service Flow Runtime passthrough map header (CIA-MAP-HEADER).

CIAMH-STRUCID

Structure Identifier. The value must be CIAMH. This is always an input field. The initial value of this field is CIAMH.

CIAMH-DATA-FORMAT-INDICATOR

This field indicates the format of the input application request or output application reply that follows this map header. Valid values are:

- I - unformatted input
- B - BMS symbolic map/application data structure (ADS) format
- 3 - 3270 data stream format
- T - text format as the result of a SEND/SEND TEXT vector

This is an input or output field. The initial value of this field is spaces. On input however, the only supported values are **I** and **B**. Meaning the service requestor can only send application request data that is unformatted (i.e., to initiate a transaction with terminal input data or 3270 data stream format data) or in an ADS format (for example, an ADS received as an application reply as the result of a SEND MAP vector).

CIAMH-MAP-DATA-LENGTH

This field indicates the length of the input application request or output application reply data that follows this map header. If CIAMH-DATA-FORMAT-INDICATOR indicates unformatted input, this field indicates the length of the unformatted application request data. If CIAMH-DATA-FORMAT-INDICATOR indicates a BMS ADS format, this field indicates the length of the BMS ADS format either in the application request or application reply data.

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

CIAMH-MAP-NAME
This field indicates the BMS map name associated with the BMS ADS data. This field applies only if CIAMH-DATA-FORMAT-INDICATOR indicates the data format is BMS ADS.

This is an input or output field. The initial value of this field is spaces.

CIAMH-MAPSET-NAME
This field indicates the BMS map set name that contains the BMS map indicated in CIAMH-MAP-NAME. This field applies only if CIAMH-DATA-FORMAT-INDICATOR indicates the data format is BMS ADS.

This is an input or output field. The initial value of this field is spaces.

CIAMH-MAP-ROWS
This field indicates the number of screen rows in the BMS map if present.

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

CIAMH-MAP-COLUMNS
This field indicates the number of screen columns in the BMS map if present.

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

CIAMH-FIELD-COUNT
This field indicates the number of fields within the ADS, that is the number of named fields in the BMS map definition macros. A separate field is counted for each element of an array defined with the OCCURS parameter but subfields of group fields (GRPNAME) are not counted. The field count may be zero, in which case there are no ADS field descriptors following this header.

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

CIAMH-EXT-ATTRIBUTE-COUNT
This field indicates the number of extended attributes in each field of the ADS, this is the number of attributes specified in the DSATTS parameter in the BMS map definition.

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

CIAMH-EXT-ATTRIBUTES
This field indicates the one character code for the attribute types in each field, in order, derived from the DSATTS parameter in the BMS map definition. Valid values are:

- C = COLOR
- P = PS
- H = HILIGHT
- V = VALIDN
- O = OUTLINE
- S = SOSI
- T = TRANSP

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

CIAMH-CURSPOS-FIELDNAME
This field indicates the name of the ADS field where the cursor is initially positioned on output, and finally positioned on input.

This is an input and output field. The initial value of this field is spaces.

Related reference:

“CIA-MAP-HEADER header structure” on page 94
When the server runtime processing is running in passthrough mode, the CIA-MAP-HEADER structure is required. It must immediately follow the CIA-SCREEN-HEADER structure. To view the passthrough map header, see copybook DFHMALMH in the SCIZMAC library.

Chapter 7. Managing adapter services

When you have successfully deployed one or more adapter services into CICS, you can use a combination of CICS commands and supplied jobs to administer them.

The properties file contains the complete record of every adapter service that has been deployed into the runtime environment. Every adapter service is also associated with a PROCESSTYPE resource.

If you are using the recommended method of defining a PROCESSTYPE resource with the same name as the request name of the modeled flow, you can use CICS commands to inquire, disable and re-enable the PROCESSTYPE resources that are defined in the runtime environment. This allows you to easily see what resources are defined, and therefore what adapter services are deployed in the runtime environment, as well as removing access temporarily or permanently to a deployed adapter service.

Otherwise you can dump the properties file to view all of the adapter services.

Disabling access to adapter services

Every adapter service is associated with a PROCESSTYPE resource. The process type can be specified in the COMMAREA or container that is passed as the request message when invoking an adapter service.

By disabling the PROCESSTYPE resource for an adapter service, you can temporarily stop that particular adapter service from being invoked in the runtime environment.

1. List all of the PROCESSTYPE resources defined in your CICS system using the **CEMT INQUIRE PROCESSTYPE** command.
2. When you have found the PROCESSTYPE resource that matches the adapter service that you want to disable, use the STATUS(DISABLED) option on the **CEMT SET PROCESSTYPE** command to disable the resource.

Any service requesters that use the process type for the disabled resource in the request message receive a CIA prefixed error message.

If you want to re-enable a PROCESSTYPE resource, use the STATUS(ENABLED) option on the **CEMT SET PROCESSTYPE** command.

Dumping the properties file

The properties file contains the definitions for all the adapter services that you have deployed in the runtime environment. To view the file, you need to dump it using a supplied sample job.

The properties file can also be used to help with troubleshooting problems. Many messages contain user response recommendations that instruct you to dump the properties file DFHMAMPF.

CICS Service Flow Runtime provides you with sample JCL to do this:

1. Check that the sample module DFHMADUP has been compiled according to the instructions in Chapter 3, “Installing the CICS Service Flow Runtime,” on page 21. The module is located in the .SCIZSAMP product distribution library.

© Copyright IBM Corp. 2001, 2010

2. Run the sample JCL job DFHMAMPD, also located in SCIZSAMP. This runs the sample module DFHMADUP , which dumps the properties file. To view the sample JCL, see “Properties file dump JCL (DFHMAMPD)” on page 267

The following example demonstrates what a properties file dump would look like.

```
02/13/06                                CICS SFR Properties file (DFHMAMPF) Dump                                PAGE
 ------------------------------------------------------------------------------------------------------------------------

  Property type:  R (Request properties)       Name:  DEMO_NAV
  Request type:  0 (Async)       Navigator name:  DEMOFNAV     Navigator transid:  DF01
  Property type:  3 (FEPI node)                Name:  DEMOBRW
  Pool name:  CICSDEV2     Target name:               Timeout value:          25  seconds   Exit action:  R (Release)
  Non-unique user:  N
  Property type:  3 (FEPI node)                Name:  DEMOINQ
  Pool name:  CICSDEV2     Target name:               Timeout value:          25  seconds   Exit action:  R (Release)
  Non-unique user:  N
  Property type:  3 (FEPI node)                Name:  DEMOSGOF
  Pool name:  CICSDEV2     Target name:               Timeout value:          25  seconds   Exit action:  Override
  Non-unique user:  N
  Property type:  3 (FEPI node)                Name:  DEMOSGON
  Pool name:  CICSDEV2     Target name:               Timeout value:          25  seconds   Exit action:  A (Leave assigned
  Non-unique user:  N
  Property type:  1 (DPL node)                 Name:  DFHMAIP2
  Linked-to program:  DFHMABP4     Remote system name:  MAD2     Mirror transaction:           SYNCONRETURN:  N
  Property type:  2 (MQSeries node (PUT))      Name:  DFHMAIP3
  MQMD MsgType:  1 (Request)      Request queue:  CIA.IVP.DFHMABP6.REQUEST.QUEUE
  MQMD ReplyToQ:  CIA.IVP.DFHMABP6.REPLY.QUEUE                    MQMD ReplyToQMgr:
  Property type:  4 (MQSeries node (GET))      Name:  DFHMAIP4
  MQGMO WaitInterval:          30  seconds
  MQMD ReplyToQ:  CIA.IVP.DFHMABP6.REPLY.QUEUE                    MQMD ReplyToQMgr:
  Property type:  3 (FEPI node)                Name:  DFHMAIP5
  Pool name:  MQIACD2      Target name:               Timeout value:          20  seconds      Exit action:  R (Release)
  Non-unique user:  N
  Property type:  5 (Link3270 bridge node)     Name:  DFHMAIP6
  Service name:  (None)               Facilitylike:           Keeptime:   3600  secs     Getwaitinterval:     1  msecs
  Deallocate on exit:  1  (Always)             Non-unique userid:  N     AOR routing:  N     Vector logging:  ON
  Property type:  R (Request properties)       Name:  L3270REQ
  Request type:  1 (Sync)        Navigator name:  TESTL327     Navigator transid:  L327
  Property type:  R (Request properties)       Name:  MAIVPREQ
  Request type:  1 (Sync)        Navigator name:  DFHMAIP1     Navigator transid:  CMA5
  Property type:  5 (Link3270 bridge node)     Name:  PROGRAM1
  Service name:  (None)               Facilitylike:           Keeptime:   3600  secs     Getwaitinterval:  4200000  msecs
  Deallocate on exit:  1  (Always)             Non-unique userid:  N     AOR routing:  N     Vector logging:  ON
  Property type:  5 (Link3270 bridge node)     Name:  PROGRAM2
  Service name:  (None)               Facilitylike:           Keeptime:   3600  secs     Getwaitinterval:  4200000  msecs
  Deallocate on exit:  1  (Always)             Non-unique userid:  N     AOR routing:  N     Vector logging:  ON
```

Figure 25. Dumped Properties file

Server runtime utilities

The following 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 can be found in the samples library *hlqual*.SCIZSAMP, that was created when you ran the set up procedure DFHMAINJ. You must compile the sample programs that are provided before they can be executed. See “Performing post-installation tasks” on page 22 for further information.

Table 9. CICS Service Flow Runtime utilities

Name	Short name	Description
Sample JCL job to run the properties file update	DFHMAMPU	Sample JCL to run load module DFHMAMUP. See “Properties file update JCL (DFHMAMPU)” on page 265 for further information.
Sample JCL job to run the Link3270 vector log file dump	DFHMAMVD	Sample JCL to run load module DFHMAVUP See “Link3270 Vector Log file dump JCL (DFHMAMVD)” on page 271 for further information.
Sample JCL job to run the error file dump	DFHMAMED	Sample JCL to run load module DFHMAEUP. See “Error file dump JCL (DFHMAMED)” on page 269 for further information.
Sample JCL job to run the properties file dump	DFHMAMPD	Sample JCL to run load module DFHMADUP. See “Properties file dump JCL (DFHMAMPD)” on page 267 for further information.
CICS Service Flow Runtime properties file update	DFHMAMUP	Provides program load module that updates the properties file DFHMAMPF. This program runs when you deploy a new adapter service. You can also run it manually using the sample JCL job, DFHMAMPU. See “Properties file dump” on page 272 for request and server adapter property value settings and descriptions.
CICS Service Flow Runtime error file dump	DFHMAEUP	Sample program that dumps the error file DFHMAERF. See “Error file dump” on page 283 for further information. See “Error processing” on page 195for further information regarding the error log facility.
CICS Service Flow Runtime properties file dump	DFHMADUP	Sample program that dumps the CICS Service Flow Runtime Properties file (DFHMAMPF). See “Properties file dump” on page 272 for request and server adapter property value settings and descriptions.
CICS Service Flow Runtime Link3270 vector log file dump	DFHMAVUP	Sample program that dumps the active Link3270 vector log file, either DFHMALVA or DFHMALVB. See “Vector file dump” on page 312 for further information.

Chapter 8. The runtime environment components

A number of programs, files and utilities make up the runtime environment. These are explained in the following chapters:

- CICS Service Flow Runtime programs
- CICS Service Flow Runtime files
- CICS Service Flow Runtime utilities

Server runtime programs

The following table lists all of the programs and describes the processing performed by each.

Table 10. CICS Service Flow Runtime programs

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime XML Header to COBOL Converter	DFHMAXMI		This program is called by the DPL Stub programs (DFHMADPL or DFHMADPP) when the request message header is in XML format. The CICS Service Flow Runtime XML Header to COBOL Converter program converts the XML into COBOL data structure so that it can be processed by the CICS Service Flow Runtime.	For a description of how this program is called, see “XML request and response processing” on page 182
CICS Service Flow Runtime COBOL to XML converter	DFHMAXMO		This program is called by the DPL stub programs (DFHMADPL or DFHMADPP) when the service requestor expects that the response to it's request should be in XML format. The CICS Service Flow Runtime COBOL to XML Converter program converts the COBOL data structure into XML so that it can be processed by the service requestor.	For a description of how this program is called, see “XML request and response processing” on page 182
CICS Service Flow Runtime ADS to XML converter	DFHMAXPI		DFHMAXPI is called by the Passthrough DPL Stub program (DFHMADPP). DFHMAXPI parses the inbound XML passthrough request message (the portion of the XML request message following the header structures, DFHMAH and DFHMAH2) and returns the passthrough application data in fixed format to the calling system program. DFHMAXPI also builds the outbound XML passthrough application reply message using any passthrough application reply data passed to it from the calling system program.	For a description of passthrough processing of an XML request, see“XML request and response processing for passthrough” on page 188

© Copyright IBM Corp. 2001, 2010

Table 10. CICS Service Flow Runtime programs (continued)

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime DPL Stub	DFHMADPL		This program defines and runs the BTS process that implements an instance of the Adapter service. The DPL Stub program also invokes the programs to support XML messaging (parsing and conversion) from the service requestor for non-passthrough requests.	Runs under the Transaction ID of the invoking program. An interface between the service requestor and CICS is used to pass a communications area to the DPL Stub program.
CICS Service Flow Runtime DPL Passthrough Stub	DFHMADPP		This program must be used when processing in passthrough mode. This program defines and runs the BTS process that implements an instance of passthrough processing. The DPL Passthrough Stub program also invokes the programs to support XML messaging (parsing and conversion) from the service requestor for passthrough requests.	CICS Service Flow Runtime DPL Passthrough Stub runs under the transaction ID of the invoking program. An interface between the service requestor and CICS is used to pass a communications area to the DPL Passthrough Stub program. If the DFHMAH-UOWCONTROL field in the DFHMAH message header is set to **3** for passthrough processing, DFHMADPP must be the stub program name passed through the interface when invoking the CICS Service Flow Runtime. If DFHMAH-UOWCONTROL is set to 3, and the Stub program name is DFHMADPL, the runtime returns an error to the service requestor.

Table 10. CICS Service Flow Runtime programs (continued)

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime Navigation Manager	DFHMAMGR	CMAM	The functions performed by the Navigation Manager program depend on the deployment pattern of the Adapter service. • When the deployment pattern is complex, requiring navigation of target applications, the Navigation Manager invokes the CICS Service Flow Runtime Navigator necessary to fulfill the request as defined by the request properties. • When the processing pattern is simple, requiring no navigation, or simple screen sequencing only, the Navigation Manager invokes the CICS Service Flow Runtime server adapter necessary to fulfill the request as defined by the request properties. • When the processing pattern is passthrough, the Navigation Manager invokes the CICS Service Flow Runtime Link3270 Passthrough Manager (DFHMALPT).	Runs as DFHROOT in all BTS processes. See “Deployment patterns” on page 53 for a description of the different processing patterns supported by the CICS Service Flow Runtime.
CICS Service Flow Runtime Navigator	User defined	User defined	Child to the Navigation Manager and parent to the server adapters. Navigators perform request processing, manage states during the service flow processing and invoke server adapters.	The Navigator program that is generated is based upon the Adapter service that was modeled and generated using the Service Flow Modeler
CICS Service Flow Runtime Error Listener (WebSphere MQSeries)	DFHMAERR	CMAE	This program monitors the defined WebSphere MQSeries error queue (CIA.SYSTEM.ERROR.QUEUE). When the Error Listener program is triggered, it reads from the error queue and writes the error to the CICS Service Flow Runtime Error file (DFHMAERF).	Error handling by the CICS Service Flow Runtime is configurable. See “Error processing” on page 195 for explanations of how errors are handled at run time.
CICS Service Flow Runtime Error Listener (CICS intrapartition transient data queue (TDQ))	DFHMAERQ	CMAQ	This program monitors the defined CICS intrapartition transient data queue (CMAQ) for errors. When the Error Listener program is triggered, it reads from the error queue and writes the error to the CICS Service Flow Runtime Error file (DFHMAERF).	Error handling by the CICS Service Flow Runtime is configurable. See “Error processing” on page 195 for explanations of how errors are handled at run time.

Table 10. CICS Service Flow Runtime programs (continued)

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime Link3270 Facility State Cleanup (Temporary storage queues (TSQ))	DFHMALSC	CMAK	The CICS Service Flow Runtime State Cleanup program (TSQ) browses the Link3270 facility state temporary storage (TS) queues and deletes expired Link3270 facility session state data and de-allocates associated Link3270 bridge facilities that CICS has not automatically deleted due to the facility being inactive for the keeptime interval.	The CICS Service Flow Runtime Facility State Cleanup is used for Link3270 Adapter services that are simple and nonpersistent. You can configure the CICS Service Flow Runtime so that this program is started automatically on CICS system initialization. For information, see “Configuring the autostart procedure for the Link3270 facility state cleanup programs” on page 30. This program performs a CICS enqueue on a resource that specifies a variable equal to the 16 byte Link3270 facility state TSQ QNAME. For a description of facility state cleanup processing on TSQs, see “Facility state cleanup processing — TSQ” on page 169
CICS Service Flow Runtime Link3270 Facility State Management (Temporary storage queues (TSQ))	DFHMALTS		The CICS Service Flow Runtime State Management (TSQ) is responsible for saving, retrieving and deleting state information from theCICS Service Flow Runtime Link3270 State TSQ for Link3270 Adapter services of the Single connector, nonpersistent type.	This is a called subprogram. It runs under the transaction ID of the generated Link3270 server adapter if performing adapter services or the transaction ID of the Link3270 Passthrough Manager (DFHMALPT) transaction ID (CMAL) if performing passthrough processing. This program performs a CICS enqueue on a resource that specifies a variable equal to the 16 byte Link3270 facility state TSQ QNAME. See “Business state data management in single connector (nonpersistent patterns)” on page 181 for further information.

Table 10. CICS Service Flow Runtime programs (continued)

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime Link3270 Facility State Cleanup (VSAM)	DFHMALFC	CMAF	This program monitors the CICS Service Flow Runtime Link3270 State VSAM file, (DFHMAL2F) and deletes expired Link3270 facility session state data and de-allocates associated Link3270 bridge facilities that CICS has not automatically deleted due to the facility being inactive for the keeptime interval.	The CICS Service Flow Runtime Facility State Cleanup (VSAM) is used for Link3270 Adapter services of the following styles: • Aggregate connector, persistent • Aggregate connector, nonpersistent • Single connector, persistent You can configure the CICS Service Flow Runtime so that this program is started automatically on CICS system initialization. For information, see “Configuring the autostart procedure for the Link3270 facility state cleanup programs” on page 30 For a description of facility state cleanup processing on the VSAM file, see“Facility state cleanup processing — VSAM” on page 169
CICS Service Flow Runtime Link3270 Facility Deallocate Cleanup	DFHMALFD	CMAD	The CICS Service Flow Runtime 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 (TSQ) or VSAM file.	Business state data is stored in a TSQ if the Adapter service is of the single connector nonpersistent type. For all other Adapter service types, business state data is stored in the CICS Service Flow Runtime Link3270 Facility State VSAM file (DFHMAL2F). For a description of facility state cleanup processing, see “Facility state cleanup processing in the CICS Service Flow Runtime” on page 168

Table 10. CICS Service Flow Runtime programs (continued)

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime Link3270 Facility State Management (VSAM).	DFHMALFS		The CICS Service Flow Runtime State Management (VSAM) program is responsible for saving, retrieving and deleting state information from the CICS Service Flow Runtime Link3270 State file (DFHMAL2F) for Link3270 Adapter services of the following types: • Aggregate connector, persistence • Aggregate connector, nonpersistent • Single connector, persistence	This is a called subprogram. It runs under the transaction ID of the generated Link3270 server adapter if performing adapter services or the transaction ID of the Link3270 Passthrough Manager (DFHMALPT) transaction ID (CMAL) if performing passthrough processing. CICS Service Flow Runtime Link3270 State file (DFHMALSF) supports Adapter services and flows from MQSI Agent for CICS that have not been regenerated in CICS Service Flow Runtime. If the adapter service or flow is built and generated using the Service Flow Modeler, the file from which state information is stored, retrieved and deleted is (DFHMAL**2**F). Although it is recommended that customers who are running Link3270 server adapters that were built and generated using MQSI Agent for CICS v1.1.3 regenerate their Adapter services for use in the CICS Service Flow Runtime, DFHMALSF provides upward compatibility.
CICS Service Flow Runtime Link3270 Passthrough Manager	DFHMALPT	CMAL	The CICS Service Flow Runtime Link3270 Passthrough Manager is run by the Navigation Manager to handle passthrough request processing. It is a child activity run by the Navigation Manager activity.	See “Passthrough processing” on page 147 for information on how this program functions when the CICS Service Flow Runtime is invoked with a passthrough request message.
CICS Service Flow Runtime Link3270 Maintenance	DFHMALNM		The CICS Service Flow Runtime Link3270 Maintenance program is invoked by both the generated Link3270 server adapter during request processing and by the Passthrough Manager program (DFHMALPT) for passthrough processing. The CICS Service Flow Runtime Link3270 Maintenance program performs initiate / terminate processing for generated Link3270 server adapters and passthrough processing.	This is a called subprogram. It runs under the transaction ID of the generated Link3270 server adapter if performing adapter services or the transaction ID of the Link3270 Passthrough Manager (DFHMALPT) transaction ID (CMAL) if performing passthrough processing..

Table 10. CICS Service Flow Runtime programs (continued)

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime FEPI 3270 Data Stream Conversion	DFHMAF		The CICS Service Flow Runtime FEPI 3270 Data Stream Conversion program is used to convert 3270 data streams to a 1920 byte record format (screen size 24 x 80). It is also used to build 3270 data streams from the 1920 byte record format.	This is a called subprogram. It runs under the transaction ID of the generated 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.
CICS Service Flow Runtime Link3270 Initiate Terminate program	DFHMALIN	CMAI	TheCICS Service Flow Runtime Link3270 Initiate Terminate program retrieves any target CICS application transaction COMMAREA information and any TCTUA information (actions performed by terminate processing) from one CICS region (region where the Link3270 bridge facility is currently allocated) and populates that same information (initiate processing) in a second CICS region before running the next target CICS application transaction routed to that second CICS region.	See “Configuring the runtime environment to use transaction routing” on page 165 for a description of how DFHMALIN is used.
CICS Service Flow Runtime Link3270 Vector Logging	DFHMAVCL		The CICS Service Flow Runtime Link3270 Vector Logging program writes Link3270 vector data to the Vector log file (DFHMALVF). It is used in both generated Link3270 server adapters and passthrough processing.	This is a called subprogram. It runs under the transaction ID of the generated Link3270 server adapter if performing adapter services or the transaction ID of the Link3270 Passthrough Manager (DFHMALPT) transaction ID (CMAL) if performing passthrough processing.

Table 10. CICS Service Flow Runtime programs (continued)

Name	Program name	CICS Tran ID	Description	Comments
CICS Service Flow Runtime Link3270 Vector Processor	DFHMAVCP		The Link3270 Vector Processor program is invoked by both the generated Link3270 server adapter during request processing, and the Passthrough Manager program (DFHMALPT) for passthrough processing. The Link3270 Vector Processor program sends vectors to and receives vectors from the Link3270 bridge program (DFHL3270) to interface to any CICS target application that uses BMS commands (with some restrictions) where a single send and receive structure inclusive of the Link3270 bridge header and appropriate input/output vector header is not greater than 32000 bytes.	This is a called subprogram. It runs under the transaction ID of the generated Link3270 server adapter if performing adapter services or the transaction ID of the Link3270 Passthrough Manager (DFHMALPT) transaction ID (CMAL) if performing passthrough processing.

CICS Service Flow Runtime utilizes the following IBM WebSphere MQ tasks when the service requestor uses WebSphere MQ to invoke the server runtime.

These components are part of the WebSphere MQ-CICS bridge. WebSphere MQ-CICS bridge is not part of CICS Service Flow Runtime, but it should be used as the interface between WebSphere MQ and CICS Service Flow Runtime when the service requestor uses WebSphere MQ to invoke the runtime.

Table 11. WebSphere MQ tasks used by CICS Service Flow Runtime

Name	Program name	CICS Tran ID	Description	Comments
WebSphere MQ-CICS bridge monitor task	CSQCBR00	CKBR	This task monitors the request queue for messages. When a message arrives in the queue, the bridge monitor task starts the WebSphere MQ CICS bridge link task.	WebSphere MQ-CICS bridge enables an application, not running in a CICS environment, to run a program or transaction in CICS and get a response back.
WebSphere MQ-CICS bridge link task	CSQCBP00	CKBP	This task pulls the message off the request queue and links to the DPL Stub program with the DFHMAH header information and the application data.	WebSphere MQ-CICS bridge enables an application, not running in a CICS environment, to run a program or transaction in CICS and get a response back.

Server runtime files

The following table lists all of the CICS Service Flow Runtime files.

Key to Table 12 on page 111:

NAME Descriptive file name.

SHORT NAME
CICS file name.

DESCRIPTION
Explanation of what the file is used for during run time processing.

FILE TYPE
VSAM File Type:

K Key sequenced data set (KSDS)

E Entry sequenced data set (ESDS)

COMMENTS
Contains CICS Service Flow Runtime processing information as it relates to the named file.

Table 12. CICS Service Flow Runtime VSAM files

Name	Short name	Description	File type	Comments
CICS Service Flow Runtime FEPI Target Interaction	DFHMATIF	Contains target (back-end target CICS application transaction) interaction information (i.e., last screen buffer sent or received using the allocated conversation for the target node).	K	This file is used only in FEPI server adapters. See “CICS Service Flow Runtime FEPI file processing” on page 157.
CICS Service Flow Runtime FEPI (SLU) Connection	DFHMACOF	Base FEPI connection file. It is used to store allocated FEPI conversation information.	K	This file is used only in FEPI server adapters. See “CICS Service Flow Runtime FEPI file processing” on page 157.
CICS Service Flow Runtime FEPI (SLU) Connection alternate	DFHMAC1F	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. See “FEPI node properties” on page 278 for further information.	K	If your site is using *LU assignment processing for non-unique UserIDs*, this file is used to establish connections and conversations by a unique tag comprised of the user id, CICS Applid and EIBTASKN of the CICS Service Flow Runtime DPL Stub program (DFHMADPL). See “Run time processing of non-unique UserIDs using FEPI adapters” on page 159 for more information.

Table 12. CICS Service Flow Runtime VSAM files (continued)

Name	Short name	Description	File type	Comments
CICS Service Flow Runtime Error	DFHMAERF	Contains CICS Service Flow Runtime processing errors. Written to by the Error Listener program. See “Error processing” on page 195 for further information.	E	If you want to view the errors, you can run the sample JCL job provided, DFHMAMED, found in the sample library, .SCIZSAMP. The sample JCL job runs the sample error dump program (DFHMAEUP) also provided in the sample library, .SCIZSAMP. This will provide a formatted error dump. You must however compile the error dump program. See “Performing post-installation tasks” on page 22 for further information. See “Server runtime utilities” on page 100 See also “Error file dump” on page 283 for further information regarding error file field definitions.
CICS Service Flow Runtime Link3270 Facility Business State	DFHMAL2F	Contains Link3270 facility business (application) state information for the allocated facility. For example; the last BMS ADS sent and the vector header information.	K	This file is used only in Link3270 server adapters. It is only used if the Link3270 Adapter services are of the following type: • Aggregate connector persistence • Aggregate connector nonpersistent • Single connector persistence See “Managing state information” on page 179 for further information.

Table 12. CICS Service Flow Runtime VSAM files (continued)

Name	Short name	Description	File type	Comments
CICS Service Flow Runtime Properties	DFHMAMPF	Contains information needed by the DPL Stub program and the server adapters in order to perform adapter request processing. Each Adapter service modeled, generated and deployed using Service Flow Modeler will have a properties file record for the associated request name and each server adapter that is part of the Adapter service.	K	The Properties file is updated by the generator facility of Service Flow Modeler automatically at build time. It can also be updated manually by running the sample JCL job provided, DFHMAMPU, found in the library .SCIZSAMP. CICS Service Flow Runtime also provides samples to provide a formatted dump of the Properties file. See “Properties file dump” on page 272 and “Server runtime utilities” on page 100 for further information regarding property values definitions and formatted dump processing.
CICS Service Flow Runtime Link3270 Facility Business State	DFHMALSF	See DFHMAL2F description above.	K	This file is used in MQSI Agent for CICS V1.1.3 Link3270 server adapter processing only. It is used by existing MQSI Agent for CICS deployed Link3270 server adapters that were modeled and deployed using builder tools that existed prior to the release of Service Flow Modeler. It is provided for upward compatibility only.

Table 12. CICS Service Flow Runtime VSAM files (continued)

Name	Short name	Description	File type	Comments
CICS Service Flow Runtime Link3270 Repository	DFHMALRF	Contains CICS target application BMS map information. For example, fields, attributes and initial values.	K	This file is used in MQSI Agent for CICS V1.1.3 Link3270 server adapter processing only. It is used by existing MQSI Agent for CICS deployed Link3270 server adapters that were modeled and deployed using builder tools that existed prior to the release of Service Flow Modeler. It is provided for upward compatibility only. See “Server runtime utilities” on page 100 for further information.
CICS Service Flow Runtime Link3270 Vector Log	DFHMALVF	Contains all Link3270 bridge vector information sent to and received from CICS target application transactions. The amount of Link3270 bridge vector information stored is configurable based on the Link3270 server adapter vector logging property setting. See “Link3270 node properties” on page 280 for further information.	K	This file is optional. It is used in Link3270 server adapter processing only. It is recommended that this file only be used in development for diagnostics and troubleshooting. Use caution when implementing vector logging in a production environment, as large amounts of data may be captured and performance may be impacted. See “Analyzing the vector log file dump” on page 207 for further information.

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.

Server runtime processing and the BTS NOCHECK option

The CICS Service Flow Runtime supports the **BTS NOCHECK** processing option.

It is important to know the following information regarding the BTS NOCHECK option as it pertains to CICS Service Flow Runtime processing.

- BTS NOCHECK option is only supported if the CICS Service Flow Runtime request property, MP-PERSISTENCE-IND, is set appropriately. For information on property settings, see “Request properties” on page 274.
- CICS Service Flow Runtime passthrough processing is always run using the BTS NOCHECK option. For more information on Passthrough processing, see “Passthrough processing” on page 147
- The CICS Service Flow Runtime BTS NOCHECK processing option should not be used if your deployed sequence flow has a compensation requirement.

  Pertinent CICS Service Flow Runtime data container information might not be available for any subsequent compensating flow. See “Determine if compensation is necessary” on page 58 and “How compensation processing works” on page 193 for further information.
- The BTS NOCHECK option specifies that no record is to be written to the repository data set to reserve the name of the process. Using the BTS NOCHECK option improves BTS performance by removing the write to the repository and its associated logging.
- If you use the BTS NOCHECK option, be aware that the error of specifying a non-unique process name no longer causes a PROCESSOR condition to be returned on the DEFINE PROCESS command. The error might not be discovered until much later when syncpoint occurs, making it much harder to debug.

CICS Service Flow Runtime data-containers for aggregate processing patterns

The following sections contain information on data-containers that CICS Service Flow Runtime uses for complex, "aggregate" processing patterns. For more information about aggregate processing patterns, see “Deployment patterns” on page 53.

- “Process data-containers - Aggregate processing pattern”
- “Adapter program data-containers - Aggregate processing pattern” on page 116
- “Server adapter data-containers - Aggregate processing pattern” on page 117
- “FEPI data-containers - Aggregate processing pattern” on page 118
- “Link3270 data-containers - Aggregate processing pattern” on page 118
- “Miscellaneous data-containers - Aggregate processing pattern” on page 120

Process data-containers - Aggregate processing pattern

Process containers are data-containers associated with a BTS process, (i.e., an instance of CICS Service Flow Runtime) initiated to perform server adapter processing.

Note: Process data-containers can also include a *Journal* container. See “Miscellaneous data-containers - Aggregate processing pattern” on page 120 for a description of the Journal container.

The following table lists the process data-containers.

Table 13. Process data-containers

Container Type	Contents and usage	Size (bytes)	Name	Comments
PROCESS	CICS Service Flow Runtime message header structure(s) in the adapter request message. Status of Navigation Manager and adapter service. Used in compensation and recovery	1,332	ADAPTER.PROCESS	No Comments.
INPUT	Application data in the adapter request message.	Variable max. 32,376	ADAPTER.INPUT	No Comments

See “How CICS Service Flow Runtime programs use data-containers” on page 124 for a list of the CICS Service Flow Runtime programs that use (read, write or read and write) the process data-containers.

Adapter program data-containers - Aggregate processing pattern

Adapter program data-containers are associated with an adapter service and adapter program-related processing. They are used to store the *state* and *status* data and information required by the Adapter program (Adapter activity).

The following table lists the adapter program data-containers used at run time.

Note: Adapter program data-containers can also include *Journal*, *Error* and *Shared Context* container types. See “Miscellaneous data-containers - Aggregate processing pattern” on page 120 for a description of these types of containers.

Table 14. Adapter program data-containers

Container Type	Contents and usage	Size (bytes)	Name	Comments
OUTPUT	Application data that will be sent in the adapter reply message Navigation Manager writes a copy of Adapter activity output container for purposes of returning adapter reply message	variable max. 32,760	ADAPTER.OUTPUT	In synchronous mode, the maximum reply length is 32,376 bytes
STATUS	Status information: • Detailed Adapter / Adapter service processing status • Error information	512	ADAPTER.STATUS	No comments

Table 14. Adapter program data-containers (continued)

Container Type	Contents and usage	Size (bytes)	Name	Comments
LOCAL CONTEXT	Application context private to Adapter activity.	Compiler limit for IBM COBOL	ADAPTER.LOCAL.C	Optional based on modeling. Used in asynchronous mode only.
ITERATION	Application work area required for modeled iterative processing.	Compiler limit for IBM COBOL	ADAPTER.ITERATE	Optional based on modeling. Used in asynchronous mode only.

See "How CICS Service Flow Runtime programs use data-containers" on page 124 for a list of the programs that use (read, write or read and write) the adapter program activity data-containers.

Server adapter data-containers - Aggregate processing pattern

Server adapter data-containers are data containers associated with a DPL server adapter and MQSeries server adapter, or both. They are used to store state and status data, inputs or outputs, or both input and outputs for server adapter program (server adapter *activity*).

The following table lists the server adapter data-containers used at run time.

Note: Server adapter data-containers can also include an *Error* container. See "Miscellaneous data-containers - Aggregate processing pattern" on page 120 for a description of the Error container.

DPL Server adapters use the **EXEC CICS LINK** command, that has a maximum COMMAREA length of 32,767 bytes. CICS Service Flow Runtime supports 32,500 bytes. The remaining 267 bytes are taken up by the message header. See the *CICS Application Program Reference* for more information on the **EXEC CICS LINK** command.

Table 15. Server adapter data-containers

Container Type	Contents and usage	Size (bytes)	Name
INPUT	Mapped input data.	variable max. 32,760	COMMAND.INPUT
OUTPUT	Output from the server adapters.	variable max. 32,760	COMMAND.OUTPUT
STATUS	Status information from server adapter, for example: • Interaction status • Error information	256	COMMAND.STATUS

See "How CICS Service Flow Runtime programs use data-containers" on page 124 for a list of the programs that use (read, write or read and write) the server adapter activity data-containers.

FEPI data-containers - Aggregate processing pattern

FEPI data-containers are data-containers associated with a FEPI server adapter. They are used to store state/status data, inputs and/or outputs for the FEPI server adapter program (FEPI server adapter *activity*).

The following table lists the FEPI data-containers used at run time.

Note: FEPI data-containers can also include *Shared Context* and *Error* container types. See "Miscellaneous data-containers - Aggregate processing pattern" on page 120 for a description of these types of containers.

Table 16. FEPI data-containers

Container Type	Contents and usage	Size (bytes)	Name	Comments
INPUT	Mapped input data.	variable max. 32,760	FEPI.INPUT	No Comments
OUTPUT	Output from the FEPI server adapter.	variable max. 32,760	FEPI.OUTPUT	No Comments
STATUS	Status information from FEPI server adapter, for example: • Interaction status • Error information	512	NAVIGATOR.STATUS	No Comments
STATE	Target and Node state information from FEPI server adapter	6,144	FEPI.STATE	Used only in case of FEPI server adapter error.

See "How CICS Service Flow Runtime programs use data-containers" on page 124 for a list of the CICS Service Flow Runtime programs that use (read, write or read and write) the FEPI server adapter activity data-containers.

Link3270 data-containers - Aggregate processing pattern

Link3270 data-containers are data-containers associated with a Link3270 server adapter. They are used to store state/status data, inputs and/or outputs for the Link3270 server adapter program (Link3270 server adapter *activity*).

The following table lists the Link3270 data-containers used at run time.

Note: Link3270 data-containers can also include *Shared Context* and *Error* container types. See "Miscellaneous data-containers - Aggregate processing pattern" on page 120 for a description of these types of containers.

Table 17. Link3270 data-containers

Container Type	Contents and usage	Size (bytes)	Name	Comments
INPUT	Mapped input data.	variable max. 32,760	LINK3270.INPUT	No Comments.
OUTPUT	Output from the LINK3270 server adapter.	variable max. 32,760	LINK3270.OUTPUT	No Comments.

Table 17. Link3270 data-containers (continued)

Container Type	Contents and usage	Size (bytes)	Name	Comments
STATUS	Status information from LINK3270 server adapter, for example: • Interaction status • Error information	512	NAVIGATOR.STATUS	No Comments.
STATE	State information from LINK3270 server adapter	variable max. 32,760	LINK3270.STATE	No comments

See “How CICS Service Flow Runtime programs use data-containers” on page 124 for a list of the CICS Service Flow Runtime programs that use (read, write or read and write) the Link3270 server adapter activity data-containers.

Web service server adapter data-containers

Web service server adapter data-containers are used to store state and status data about the adapter itself, as well as input and output data for the Web service request when it is invoked by the Adapter Navigator.

The following table lists the data-containers used at runtime. To see details of the ERROR type container, see “Miscellaneous data-containers - Aggregate processing pattern” on page 120.

Table 18. Web service server adapter data-containers

Name	Type	Size (bytes)	Contents and usage
WEBSERVICE.DATA	DATA		Web service request that is passed to the requester pipeline.
COMMAND.INPUT	INPUT	variable max. 32,760	Mapped input data.
COMMAND.OUTPUT	OUTPUT	variable max. 32,760	Output from the server adapters.
COMMAND.STATUS	STATUS	256	Status information from the server adapter, for example: • Interaction status • Error information

WMQ server adapter data-containers

WMQ server adapter data-containers are used to store state and status data, as well as input and outputs for the server adapter program when it is invoked by an Adapter Navigator.

The following table lists the data-containers used at run time. To see details of the ERROR type container, see “Miscellaneous data-containers - Aggregate processing pattern” on page 120.

Table 19. WMQ server adapter data-containers

Container Type	Contents and usage	Size (bytes)	Name
INPUT	Mapped input data.	variable max. 32,760	COMMAND.INPUT

Table 19. WMQ server adapter data-containers (continued)

Container Type	Contents and usage	Size (bytes)	Name
OUTPUT	Output from the server adapters.	variable max. 32,760	COMMAND.OUTPUT
STATUS	Status information from server adapter, for example: • Interaction status • Error information	256	COMMAND.STATUS

Miscellaneous data-containers - Aggregate processing pattern

Because error processing, journaling and sharing application data might be necessary at different stages of run time processing, the *Error*, *Journal* and *Shared Context* data-containers are listed together.

Table 20. Miscellaneous data-containers

Container Type	Contents and usage	Size (bytes)	Name	Comments
ERROR	Detailed error information that can be used in problem determination.	256	ADAPTER.ERROR	No Comments.
JOURNAL	Application data modeled at build time. Used in compensation and recovery by Adapter service. Navigation Manager writes a copy of Adapter activity Journal container for purposes of compensation and recovery.	4MB	ADAPTER.JOURNAL	During compensation, the DPL Stub program (DFHMADPL) writes a copy of this container to a process container for use in the compensating sequence flow.
SHARED CONTEXT	Application context shared between Adapter activity and child FEPI server adapter or child Link3270 server adapter.	Compiler limit for IBM COBOL	ADAPTER.SHARED.C	Optional based on modeling. Separate containers for each Adapter activity and FEPI server adapter or Link3270 server adapter, but mappings are the same and data is kept consistent.

See “How CICS Service Flow Runtime programs use data-containers” on page 124 for a list of the CICS Service Flow Runtime programs that use (read, write or read and write) these miscellaneous data-containers.

Data-containers for single connector processing patterns

The following sections describe the data-containers that CICS Service Flow Runtime uses for server adapters that have single connector processing patterns.

In a single connector processing pattern there is no deployed Adapter program (Adapter *activity*). Therefore there are no Adapter program data-containers associated with an Adapter activity. Deployed WebSphere MQSeries and Web services server adapters are not supported in a single connector processing pattern.

- "Process data-containers - single connector processing pattern"
- "Server adapter data containers - single connector processing pattern" on page 122
- "FEPI data-containers - single connector processing pattern" on page 122
- "Link3270 data-containers - single connector processing pattern" on page 123
- "Miscellaneous data-containers - single connector processing pattern" on page 124

Process data-containers - single connector processing pattern

Process containers are data-containers associated with a BTS process, (i.e., an instance of CICS Service Flow Runtime) initiated to perform server adapter processing.

Note: Process data-containers can also include a *Journal* container. See "Miscellaneous data-containers - single connector processing pattern" on page 124 for a description of the Journal container.

The following table lists the process data-containers.

Table 21. Process data-containers - single connector pattern

Container Type	Contents and usage	Size (bytes)	Name	Comments
PROCESS	CICS Service Flow Runtime message header structure(s) in the adapter request message. Status of Navigation Manager and adapter service. Used in compensation and recovery	1,332 for CICS Service Flow Runtime 1,640 for CICS Service Flow Runtime for passthrough mode	ADAPTER.PROCESS	No comments.
INPUT	Application data in the adapter request message.	Variable max. 32,376	ADAPTER.INPUT	No comments

See "How CICS Service Flow Runtime programs use data-containers" on page 124 for a list of the programs that use (read, write or read and write) the process data-containers.

Server adapter data containers - single connector processing pattern

For a single connector processing pattern, server adapter data containers are containers associated with a DPL server adapter only. They are used to store state and status data, inputs or outputs, or both input and outputs for the DPL server adapter activity.

The following table lists the server adapter data-containers used at run time.

Note: Deployed WebSphere MQSeries server adapters are not supported in a single connector processing pattern.

DPL Server adapters use the **EXEC CICS LINK** command, that has a maximum COMMAREA length of 32,767 bytes. CICS Service Flow Runtime supports 32,500 bytes. The remaining 267 bytes are taken up by the message header. See the *CICS Application Program Reference* for more information on the **EXEC CICS LINK** command.

The following table lists the DPL server adapter data containers used at run time.

Table 22. Server adapter data containers - single connector patterns

Container Type	Contents and usage	Size (bytes)	Name
OUTPUT	Application data that will be sent in the adapter reply message. Navigation Manager writes a copy of server adapter activity output container for purposes of returning adapter reply message.	variable max. 32,500	ADAPTER.OUTPUT
STATUS	Status information from DPL server adapter, for example • Interaction status • Error information	256	COMMAND.STATUS

See “How CICS Service Flow Runtime programs use data-containers” on page 124 for a list of the CICS Service Flow Runtime programs that use (read, write or read and write) the DPL server adapter activity data containers.

FEPI data-containers - single connector processing pattern

FEPI data-containers are containers associated with a FEPI server adapter. They are used to store state and status data, inputs and outputs for the FEPI server adapter program (FEPI server adapter *activity*).

The following table lists the FEPI data-containers used at run time.

Note: FEPI data-containers can also include *Shared Context* and *Error* container types. See “Miscellaneous data-containers - single connector processing pattern” on page 124 for a description of these types of containers.

Table 23. FEPI data-containers - single connector pattern

Container Type	Contents and usage	Size (bytes)	Name	Comments
OUTPUT	Application data that will be sent in the adapter reply message	variable max. 32,760	ADAPTER.OUTPUT	No Comments
STATUS	Status information from FEPI server adapter, for example: • Interaction status • Error information	512	ADAPTER.STATUS	No Comments
STATE	Target and Node state information from FEPI server adapter	6,144	FEPI.STATE	Used only in case of FEPI server adapter error.

See “How CICS Service Flow Runtime programs use data-containers” on page 124 for a list of the CICS Service Flow Runtime programs that use (read, write or read and write) the FEPI server adapter activity data-containers.

Link3270 data-containers - single connector processing pattern

Link3270 data-containers are containers associated with a Link3270 server adapter. They are used to store state and status data, inputs and outputs for the Link3270 server adapter program (Link3270 server adapter *activity*).

The following table lists the Link3270 data-containers used at run time.

Note: Link3270 data-containers can also include *Shared Context* and *Error* container types. See “Miscellaneous data-containers - single connector processing pattern” on page 124 for a description of these types of containers.

Table 24. Link3270 data-containers - single connector patterns

Container Type	Contents and usage	Size (bytes)	Name	Comments
OUTPUT	Application data that will be sent in the adapter reply message. Navigation Manager writes a copy of server adapter activity output container for purposes of returning adapter reply message.	variable max. 32,760	ADAPTER.OUTPUT	No Comments.
STATUS	Status information from LINK3270 server adapter, for example: • Interaction status • Error information	512	ADAPTER.STATUS	No Comments.

Table 24. Link3270 data-containers - single connector patterns (continued)

Container Type	Contents and usage	Size (bytes)	Name	Comments
STATE	State information from LINK3270 server adapter. Navigation Manager reads this container to load DFHMAH header information in reply message.	variable max. 32,760	LINK3270.STATE	No comments

See “How CICS Service Flow Runtime programs use data-containers” for a list of the programs that use (read, write or read and write) the Link3270 server adapter activity data-containers.

Miscellaneous data-containers - single connector processing pattern

Because error processing, journaling and sharing application data may be necessary at different stages of run time processing, the *Error* and *Journal* data-containers are listed together.

Table 25. Miscellaneous data-containers - single connector patterns

Container Type	Contents and usage	Size (bytes)	Name	Comments
ERROR	Detailed error information that can be used in problem determination.	256	ADAPTER.ERROR	No Comments.
JOURNAL	Application data modeled at build time. Used in compensation and recovery by Adapter service. Navigation Manager writes a copy of Adapter activity Journal container for purposes of compensation and recovery.	4MB	ADAPTER.JOURNAL	During compensation, the DPL Stub program (DFHMADPL) writes a copy of this container to a process container for use in the compensating sequence flow.

See “How CICS Service Flow Runtime programs use data-containers” for a list of the programs that use (read, write or read and write) these miscellaneous data-containers.

How CICS Service Flow Runtime programs use data-containers

The following rules apply to data-container usage at run time:

- Process containers can be read by all the activities that comprise a process but written to and updated by only the defining or acquiring program and the root activity of the process, DFHROOT.

- Each adapter program data-container can be read and updated by the owning activity, by the activity's parent, or by a program that has acquired the activity. These containers are used by the Navigation Manager and Navigators.
- Each server adapter program data-container can be read and updated by the activity itself, by the activity's parent (Adapter program activity), or by a program that has acquired the activity.
- Each FEPI server adapter program data-container can be read and updated by the activity itself, by the activity's parent (Adapter program activity), or by a program that has acquired the activity.
- Each Link3270 server adapter program data-container can be read and updated by the activity itself, by the activity's parent (Adapter program activity), or by a program that has acquired the activity.
- The Error, Journal and Shared Context containers can be characterized as one of the various CICS Service Flow Runtime data-container types, depending on how and when it is used. The Error and Shared Context containers are never process containers. Each activity implementing an Adapter service program, can have its own error container.

Chapter 9. Server runtime processing

Synchronous and asynchronous processing

Requests in the CICS Service Flow Runtime can be processed in synchronous mode or in asynchronous mode. You can use any of the supported interfaces to run an adapter service synchronously. However, to process requests asynchronously, you must use WMQ and the Websphere MQ-CICS bridge to invoke an adapter service.

Synchronous mode

When the service requester invokes an adapter service synchronously, the DFHMADPL stub program defines the BTS process for the navigation manager and runs it synchronously. The navigation manager then waits for the server adapter request processing to complete before returning to DFHMADPL. When the BTS process completes, the unit of work is committed, even if there are system errors. If you do not want this to happen, you can use the *synchronous rollback* mode

Synchronous rollback is indicated by the request record of the adapter service that is defined in the properties file. A value of 2 for the **PARM01** parameter in the Request type indicates that the request should be processed in synchronous rollback mode.

If you use synchronous rollback for your adapter service, if a failure occurs in any activity within the BTS process, an abend occurs. This has the effect of returning the state of any and all recoverable resources updated during server adapter request processing to the state it was in prior to the start of the failed request. For example, if your adapter service comprises a series of DPL requests that need to run as a single unit of work and be committed, if any one of the DPL requests is unsuccessful the unit of work is not committed.

If the navigation manager detects an error, it issues the following command:

```
EXEC CICS ABEND ABCODE('CIA')
                CANCEL
                NODUMP
END-EXEC
```

This command ends the process with abend code CIA, and causes no dump to be taken in the CICS region.

Asynchronous mode

The service requester must be a WebSphere MQ enabled application in order to invoke an adapter service asynchronously. The following scenario explains how the request message is processed asynchronously.

1. A message is sent to a queue monitored by the WebSphere MQ-CICS bridge monitor task (CKBR). The ReplyToQ and ReplyToQMgr are not loaded in MQMD Message Descriptor, but instead are included in the DFHMAH message header.
2. The bridge monitor task CKBR starts the bridge link task CKBP.
3. The bridge link task pulls the message from the queue and links to the DPL stub program DFHMADPL with the request message.

© Copyright IBM Corp. 2001, 2010

4. DFHMADPL initiates the BTS process to start the navigation manager asynchronously, using the command **BTS RUN ACQPROCESS ASYNCHRONOUS**, and returns to the bridge task. The bridge task also returns, committing the **GET** of the request message.
5. The navigation manager initiates the generated navigator asynchronously, using the **BTS RUN ACTIVITY ASYNCHRONOUS** command, and returns and waits for the navigator to complete its processing.
6. The navigator initiates the server adapters that comprise the adapter service asynchronously.
7. The navigator handles the completion events of the server adapters until all of the request processing is complete.
8. When the processing is complete, the navigation manager is invoked with the completion event of the navigator.
9. The navigation manager reads any reply message from the navigator's output data-container, and puts the reply message on the queue defined by the ReplyToQ and ReplyToQMgr fields in the DFHMAH message header.
10. The navigation manager ends the process.

Processing patterns

The processing patterns support the deployment patterns that are used to generate the Adapter services.

Single connector — persistent and nonpersistent

The following characteristics apply to CICS Service Flow Runtime processing of an Adapter service of the Single connector type:

- CICS BTS NO CHECK is implemented on **DEFINE PROCESS** command for an Adapter service of the Single connector nonpersistent type
- CICS BTS NO CHECK option is omitted on **DEFINE PROCESS** command for an Adapter service of the Single connector persistent type
- No Adapter Navigator is generated, deployed or executed as part of the modeled Adapter service.
- The Navigation Manager, DFHMAMGR, runs the appropriate server adapter as defined by the request properties.
- Only one generated and deployed FEPI, Link3270, or DPL server adapter is executed to perform the Adapter service. WebSphere MQ server adapters are not supported in the Single connector — persistent and nonpersistent pattern.
- It can provide simple "screen-sequencing" for FEPI and Link3270 Navigators with the option to run a compensating flow if failure occurs.

Request processing patterns for Single connector — persistent and nonpersistent

All processing in the single connector pattern is the same whether it is run "persistent" or "nonpersistent" with the exception of the **BTS DEFINE PROCESS** command in the CICS Service Flow Runtime DPL stub program (DFHMADPL).

The following information describes the request processing that is associated with adapter services of the single connector type.

1. A service requester initiates the CICS Service Flow Runtime when it sends a correctly formatted request message using any of the supported interfaces.

2. The DPL stub program (DFHMADPL) receives the request through one of the supported interfaces and reads the properties file DFHMAMPF, using the request name that was passed in as part of the request message in the message header structure (DFHMAH).

   See “Properties file record descriptor (DFHMARPF)” on page 268 to view the structure of the properties file.

   See “Request properties” on page 274 for request property value settings and descriptions.

   Note: If the request message is in XML format, the DPL stub program will call the XML Header to COBOL Converter program (DFHMAXMI) to parse the XML header tag data and return in the COBOL fixed format structure (DFHMAH). This call is made before reading the properties file.

   The CICS Service Flow Runtime DPL stub program calls a user-defined program to parse the XML application request tag data and return in a COBOL fixed format structure. The application program name is determined by the request properties file field, **MP-XML-PARSE-PROGRAMID**. See “Request properties” on page 274 for further information.

   See “XML request and response processing” on page 182 and “XML message formats for non-passthrough” on page 322 for further information regarding XML message processing.
3. If the request properties record indicates that the processing pattern is of the single connector type (see “Request properties” on page 274 for request property value settings and descriptions); **MP-DEPLOYMENT-IND = 1**, the DPL stub program (DFHMADPL) performs the following processing steps:
   a. Defines the BTS process using the BTS command **EXEC CICS DEFINE PROCESS**:

      For **MP-PERSISTENCE-IND = 0** (nonpersistent):

      ```
      EXEC CICS DEFINE PROCESS (processname)
                  PROCESSTYPE (processtype)
                     TRANSID (transid)
                    PROGRAM  (program)
                    NOCHECK
                       RESP (CICS-RESP)
                      RESP2 (CICS-RESP2)
      END-EXEC.
      ```

      For **MP-PERSISTENCE-IND =1** (persistent):

      ```
      EXEC CICS DEFINE PROCESS (processname)
                  PROCESSTYPE (processtype)
                     TRANSID (transid)
                    PROGRAM  (program)
                      RESP (CICS-RESP)
                      RESP2 (CICS-RESP2)
      END-EXEC.
      ```

      where:

      - (*processname*) is equal to value passed in the request message header structure, **DFHMAH**, field **DFHMAH-PROCESSNAME** or a unique value generated by CICS Service Flow Runtime. See “Request message headers” on page 80 for further information.
      - (*processtype*) is equal to value passed in the request message header structure, **DFHMAH**, field **DFHMAH-PROCESSTYPE**. See “Request message headers” on page 80 for further information.

- (*transid*) is equal to Navigation Manager transaction ID **(CMAM**).
- (*program*) is equal to Navigation Manager program name (**DFHMAMGR**). The Navigation Manager runs as the BTS root activity (DFHROOT).

The NOCHECK processing option indicates that no record is to 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 write to the repository and its associated logging. See *CICS Business Transaction Services* manual for further information on the **DEFINE PROCESS** command and options.

b. Writes application request data to the process input data-container **ADAPTER.INPUT**. The Server adapter program activity reads the input data-container **ADAPTER.INPUT** to retrieve the application request data that is used in Adapter service processing.

c. Writes CICS Service Flow Runtime state information to the process data-container **ADAPTER.PROCESS**

   State information includes the request name (**DFHMAH-REQUESTNAME**) as specified in the message header (**DFHMAH**) and the program name and transaction that will process the application request data as defined by request properties, **MP-INITIAL-PROGRAMID** and **MP-INITIAL-TRANSID** fields, respectively.

   For Adapter services of the single connector type, the program and transaction defined and run **will not be an Adapter Navigator** but rather will be a DPL, FEPI or Link3270 server adapter.

d. Runs the BTS process either in synchronous or asynchronous mode as defined by request property, **MP-REQUESTTYPE**.

   Note: If you are using a synchronous interface (i.e., ECI, EXI or DPL) to invoke CICS Service Flow Runtime processing, and if the service requestor expects a response, your Adapter service request must be run in synchronous mode. Asynchronous mode processing assumes a WebSphere MQ interface.

   Synchronous BTS RUN command (MP-REQUESTTYPE = 1 or 2)

```
EXEC CICS RUN ACQPROCESS
           SYNCHRONOUS
                   RESP (CICS-RESP)
                  RESP2 (CICS-RESP2)
END-EXEC.
```

   Asynchronous BTS RUN command (MP-REQUESTTYPE = 0)

```
EXEC CICS RUN ACQPROCESS
            ASYNCHRONOUS
                    RESP (CICS-RESP)
                   RESP2 (CICS-RESP2)
END-EXEC.
```

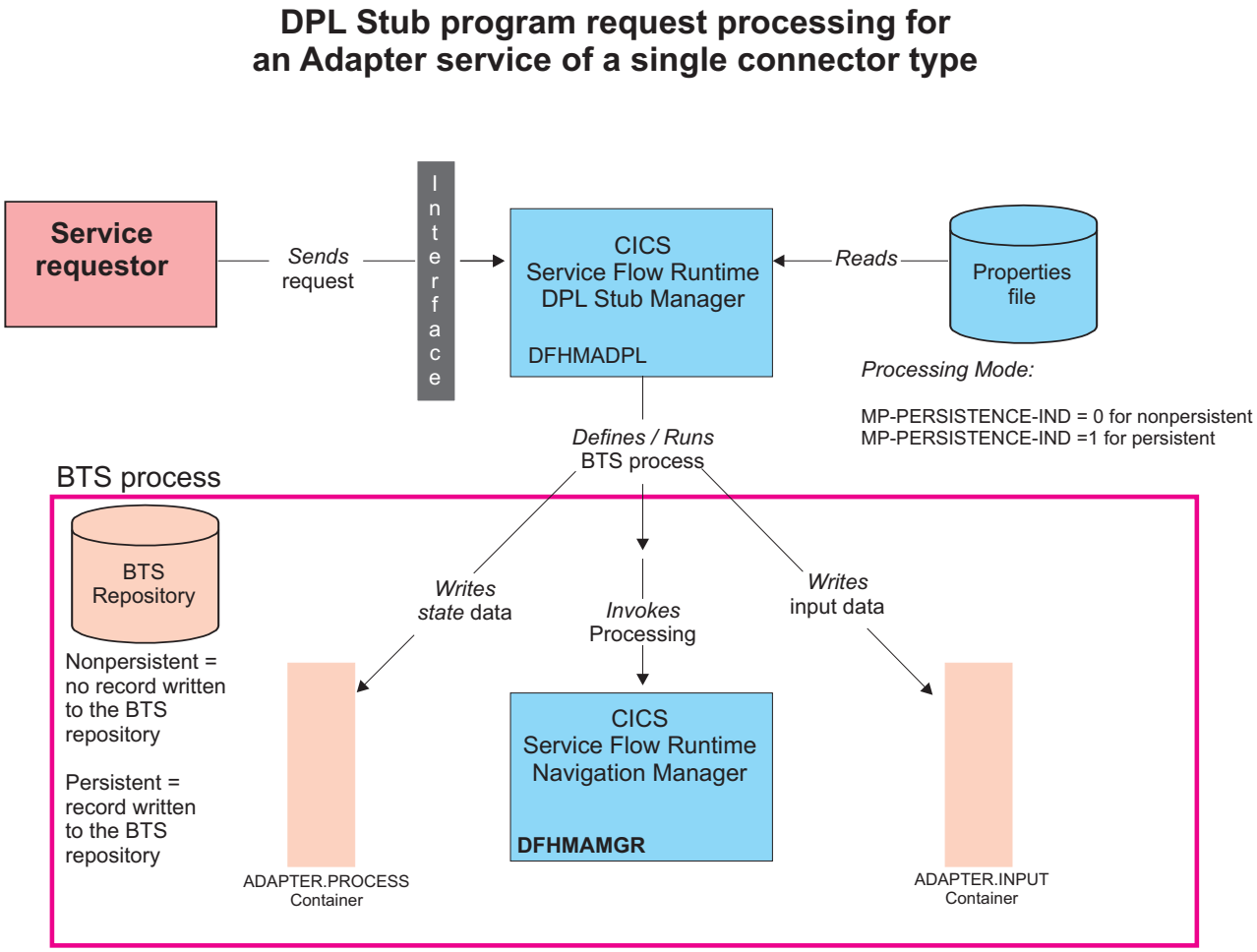


Figure 26. DPL stub program request processing — single connector type

4. Once initiated, the Navigation Manager (DFHMAMGR) performs the following processing steps:
   a. Reads the state information from process container **ADAPTER.PROCESS**.
   b. Defines the Adapter service BTS activity that will process the application request via the BTS command; **EXEC CICS DEFINE ACTIVITY**:

```
EXEC CICS DEFINE ACTIVITY  (ADC-NAV-ACTIVITY)
                    EVENT  (<completionevent>)
                  TRANSID  (ADC-NAV-TRANSACTION)
                  PROGRAM  (ADC-NAV-PROGRAM-ID)
               ACTIVITYID  (ADC-NAV-ACTIVITYID)
                     RESP  (CICS-RESP)
                     RESP2  (CICS-RESP2)
END-EXEC.
```

   where:

   - **ADC-NAV-ACTIVITY** is equal to the request name value passed and stored in the process container **ADAPTER.PROCESS**.
   - **ADC-NAV-TRANSACTION** is equal to the initial transaction ID as indicated on the request properties, field **MP-INITIAL-TRANSID**, and stored in process container **ADAPTER.PROCESS**
   - **ADC-NAV-PROGRAM-ID** is equal to the initial program name as indicated on the request properties, field **MP-INITIAL-PROGRAMID**, and also stored in process container **ADAPTER.PROCESS**.

c. Writes adapter state information to container **ADAPTER.PROCESS**.

   Additional state information includes the ACTIVITYID, CICS applid where the Navigation Manager (DFHMAMGR) is running, EIBTASKN of the running Navigation Manager, etc.

   For Adapter services of the single connector type, the program and transaction invoked **will not be an Adapter Navigator** but rather will be a DPL, FEPI or Link3270 server adapter. Only one server adapter type will be invoked for this processing pattern as defined by the Adapter service.

d. Runs the Adapter service BTS activity either in synchronous or asynchronous mode, as defined by request property, **MP-REQUESTTYPE**.

   Synchronous BTS RUN command (MP-REQUESTTYPE = 1 or 2)

```
EXEC CICS RUN ACTIVITY (ADC-NAV-ACTIVITY)
          SYNCHRONOUS
                  RESP (CICS-RESP)
                 RESP2 (CICS-RESP2)
END-EXEC.
```

   Asynchronous BTS RUN command (MP-REQUESTTYPE = 0)

```
EXEC CICS RUN ACTIVITY (ADC-NAV-ACTIVITY)
          ASYNCHRONOUS
                  RESP (CICS-RESP)
                 RESP2 (CICS-RESP2)
END-EXEC
```

Navigation Manager request processing of an Adapter Service of a single connector type

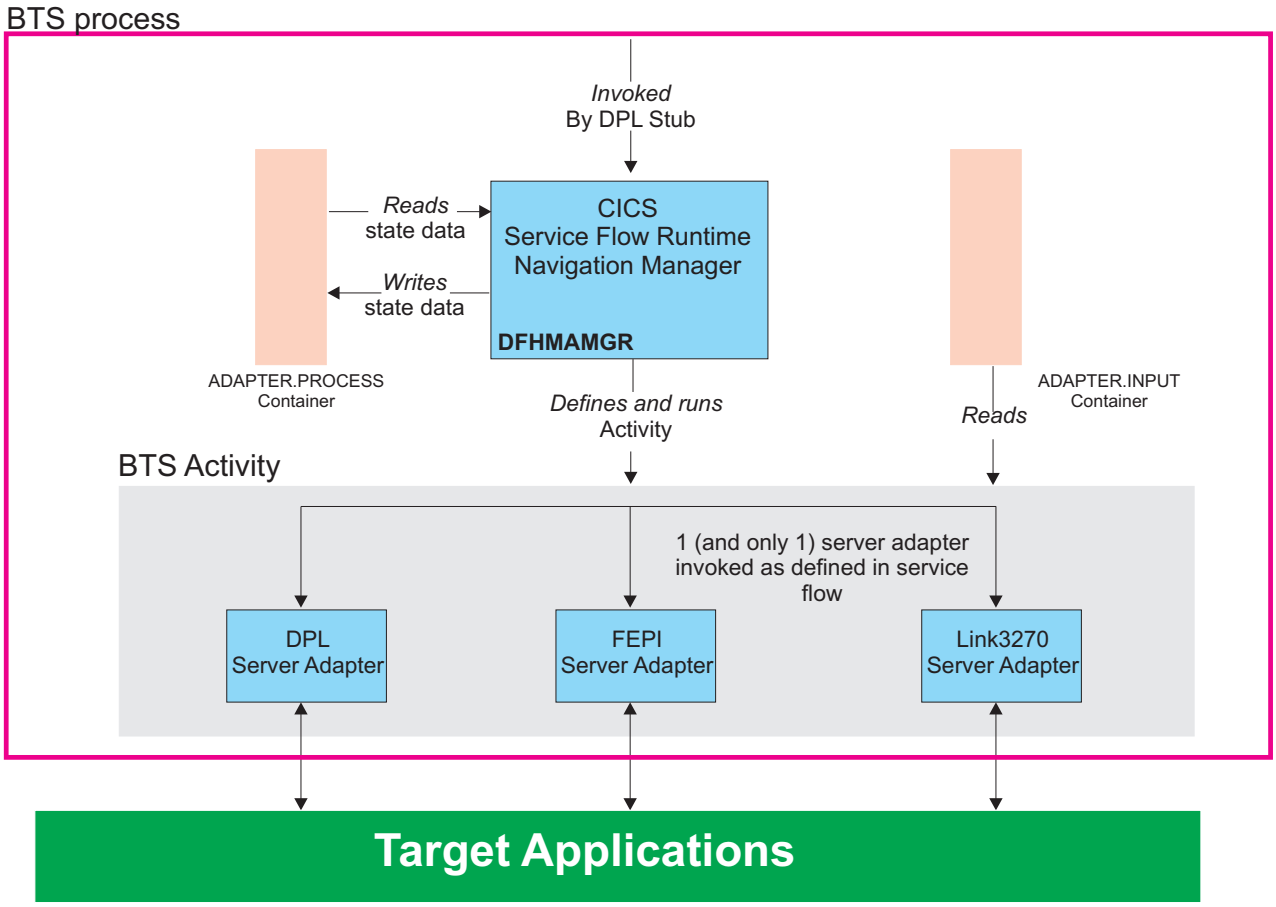


Figure 27. Navigation Manager program request processing — single connector type

As a result of the Navigation Manager (DFHMAMGR) running the Adapter service activity, the DPL, FEPI or Link3270 server adapter performs business request processing (as modelled in Service Flow Modeler) in order to complete the request made by the service requestor.

Each business request issued by the service requestor results in a different BTS process instance to fulfill the appropriate Adapter service as defined by the business request name.

Each process instance consists of a Navigation Manager (DFHMAMGR) activity and *the one server adapter activity that is needed to support that Adapter service.*

Reply processing patterns for Single connector — persistent and nonpersistent

Upon server adapter completion, any application response/reply data is written to a server adapter output data container **ADAPTER.OUTPUT**.

- DPL server adapter status information is written to a status container, name = **COMMAND.STATUS**.

- FEPI and Link3270 server adapter status information is written to a status container, name = **ADAPTER.STATUS**.

Once the server adapter has completed processing and the Adapter service activity has ended normally, a BTS completion event is fired and control is returned to the Navigation Manager (DFHMAMGR).

The Navigation Manager performs the following reply processing steps:

1. Performs a check on the server adapter activity completion status by issuing the BTS **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 state information from process container **ADAPTER.PROCESS**.
3. Reads the appropriate server adapter status information from either container **ADAPTER.STATUS** or **COMMAND.STATUS** depending upon server adapter type run.
4. Reads the server adapter output container, if normal completion, **ADAPTER.OUTPUT**.
5. Writes the root activity output data container containing the application response/reply data. The output container name is **ADAPTER.OUTPUT**.
6. Writes adapter completion status and state information to process container **ADAPTER.PROCESS**.
7. Optionally issues MQPUT if the BTS process was executed in asynchronous mode and the client application interface was WebSphere MQ-CICS bridge. Otherwise, the Navigation Manager (DFHMAMGR) issues an EXEC CICS RETURN ENDACTIVITY command returning control to the DPL stub program (DFHMADPL) and completing the process.

Navigator Manager reply processing for an Adapter service of a single connector type

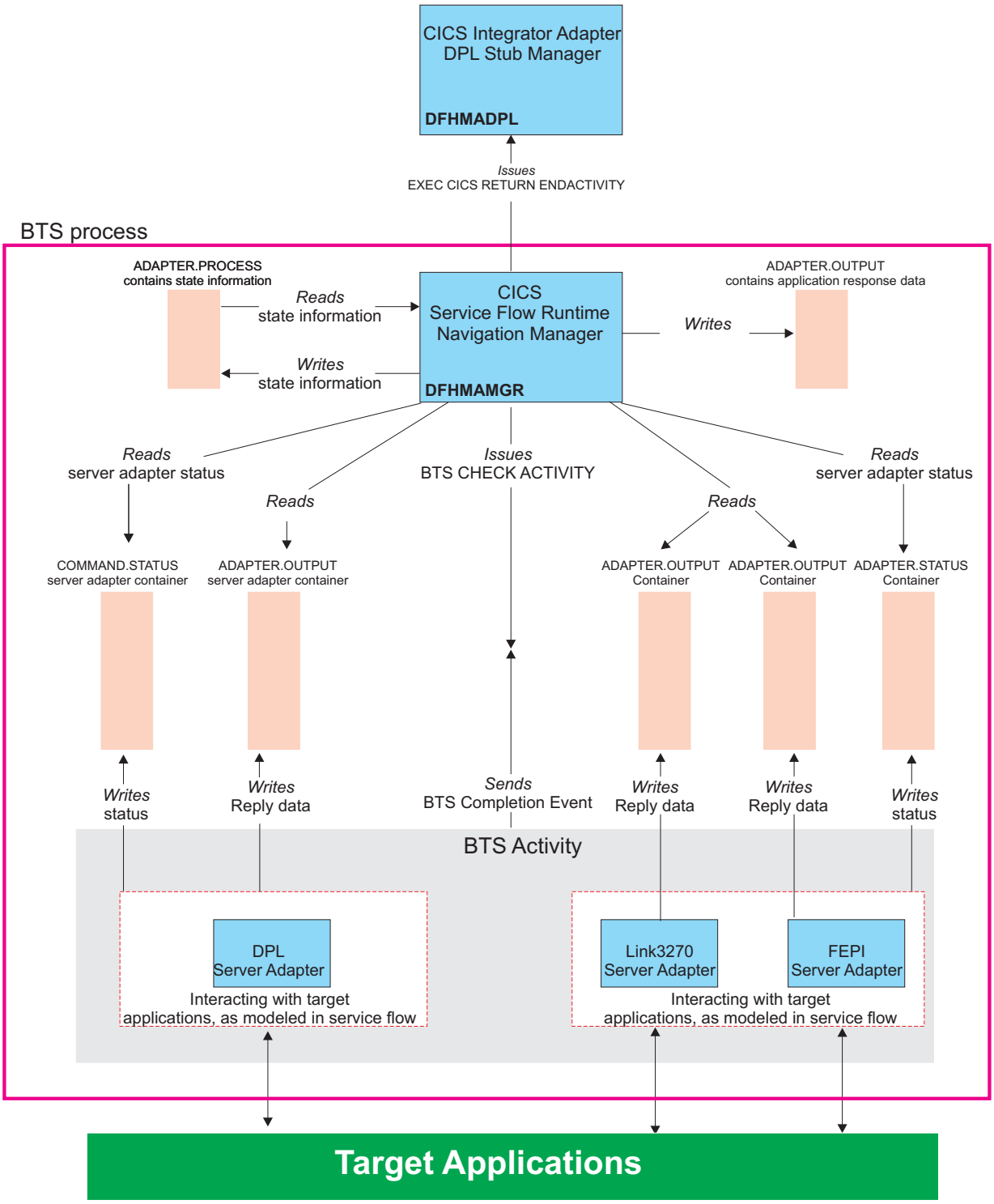


Figure 28. Navigation Manager program reply processing — single connector type

Once the Navigation Manager (DFHMAMGR) has completed processing and the BTS process has ended normally, a BTS completion event is fired and control is returned to the DPL stub program (DFHMADPL).

The DPL stub performs the following processing steps:

1. Performs a check on the process completion status by issuing the BTS **CHECK ACQPROCESS** command.
2. Reads the adapter state information from process container **ADAPTER.PROCESS**
3. Reads the Navigation Manager (DFHMAMGR) output data container **ADAPTER.OUTPUT**.
4. Builds XML or fixed format reply message to include the header structure (**DFHMAH**) and any output data container application response/reply data.

   The DPL stub program (DFHMADPL) might call a user-defined program to convert the COBOL fixed format outbound application reply structure to XML application reply tag data. The application program name is determined by the Adapter service.

   The DPL stub program will then call the CICS Service Flow Runtime COBOL to XML Converter program (DFHMAXMO) to convert the COBOL fixed format header structure (**DFHMAH**) to XML header tag data to format the XML reply message.

   See “XML request and response processing” on page 182 and “XML message formats for non-passthrough” on page 322 for further information regarding XML message processing. The Navigation Manager program (DFHMAMGR) also performs XML reply message processing as described above when the Adapter service is run in asynchronous mode and the client application interface was WebSphere MQ-CICS bridge.
5. Moves outbound reply message to **DFHCOMMAREA** and issues an EXEC CICS RETURN command returning control to the service requestor.

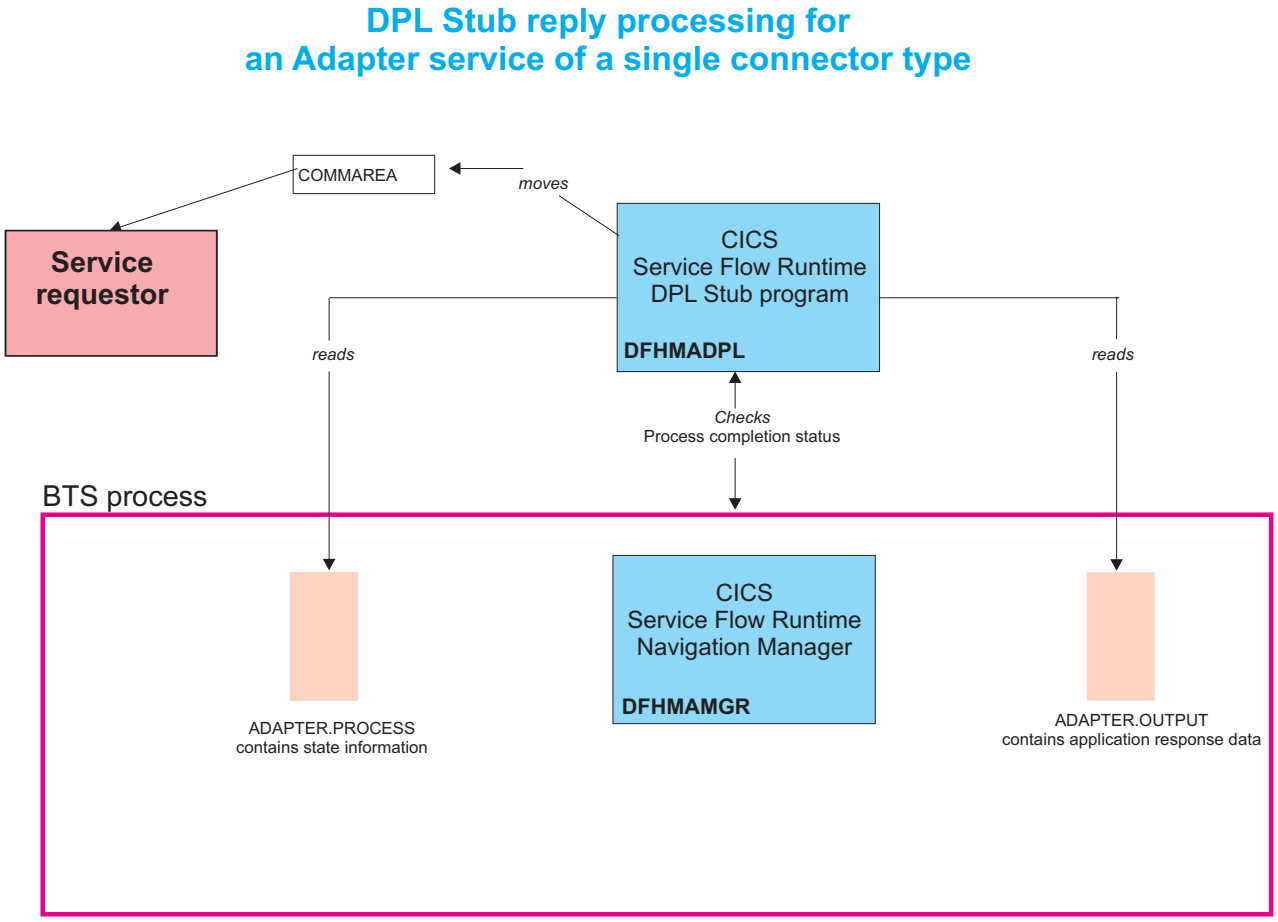


Figure 29. DPL stub program reply processing — single connector type

Aggregate connector — persistent and nonpersistent

The following characteristics apply to CICS Service Flow Runtime processing of an Adapter service of the *aggregate connector* type.

- CICS BTS NOCHECK option is implemented on **DEFINE PROCESS** command for an Adapter service of the Aggregate connector -- nonpersistent type.
- CICS BTS NOCHECK option is omitted on **DEFINE PROCESS** command for an Adapter service of the Aggregate connector -- persistent type.
- An Adapter Navigator is generated, deployed and executed as modeled in Service Flow Modeler to provide Adapter service flow navigation (i.e., control of the execution sequence of the generated and deployed server adapters that constitute the Adapter service).
- The Navigation Manager, DFHMAMGR, runs the appropriate Adapter Navigator as defined by the request properties.
- 1 to *n* number of generated and deployed FEPI, Link3270, WebSphere MQ or DPL server adapters executed as defined by the Adapter service.
- Provides for more complex processing requirements where multiple server adapters of any type may be executed to run target applications potentially to multiple back-end systems via any of the CICS Service Flow Runtime supported interfaces (FEPI, Link3270, DPL or WebSphere MQ).

- Can provide simple "screen-sequencing" for FEPI and Link3270 server adapters with option to run compensating flow if failure occurs.

Request processing patterns for Aggregate connector — persistent and nonpersistent

All processing in the aggregate connector pattern is the same whether it is run with the persistent or nonpersistent option with the exception of the BTS **DEFINE PROCESS** command in the DPL stub program (DFHMADPL).

The following information describes the request processing that is associated with Adapter services of the *aggregate connector* type:

1. A service requestor initiates the CICS Service Flow Runtime when it sends a correctly formatted request using any of the supported interfaces.
2. The DPL stub program (DFHMADPL), receives the request through one of the supported interfaces and reads the properties file (DFHMAMPF) using the request name (**DFHMAH-REQUESTNAME**) that was passed in as part of the request message in the message header structure (DFHMAH).

   See “Properties file record descriptor (DFHMARPF)” on page 268 to view the structure of the Properties file.

   See “Request properties” on page 274 for request property value settings and descriptions. If the request message is in XML format, the DPL stub program will call the XML Header to COBOL Converter program (DFHMAXMI) to parse the XML header tag data and return in a COBOL fixed format (DFHMAH). This call is made before reading the Properties file.

   The DPL stub program will then call a user-defined program to parse the XML application request tag data and return in a COBOL fixed format structure. The application program name is determined by the request properties file field, **MP-XML-PARSE-PROGRAMID**. See “Request properties” on page 274 for further information.

   See “XML request and response processing” on page 182 and “XML message formats for non-passthrough” on page 322 for further information regarding XML message processing.
3. When the request properties record indicates that the processing pattern is of the aggregate connector type (see “Request properties” on page 274 for request property value settings and descriptions); **MP-DEPLOYMENT-IND = 2**, the DPL stub program (DFHMADPL) performs the following processing steps:

   a. Defines the BTS process using the BTS command **EXEC CICS DEFINE PROCESS**:

   For **MP-PERSISTENCE-IND = 0** (nonpersistent):

```
EXEC CICS DEFINE PROCESS (<processname>)
             PROCESSTYPE (<processtype>)
                 TRANSID (<transid>)
                PROGRAM  (<program>)
               NOCHECK
                     RESP (CICS-RESP)
                    RESP2 (CICS-RESP2)
END-EXEC.
```

   For **MP-PERSISTENCE-IND =1** (persistent):

```
EXEC CICS DEFINE PROCESS (<processname>)
             PROCESSTYPE (<processtype>)
                 TRANSID (<transid>)
```

```
                PROGRAM  (<program>)
                    RESP (CICS-RESP)
                   RESP2 (CICS-RESP2)
END-EXEC.
```

where:

- (**<processname>**) is equal to value passed in the request message header structure, **DFHMAH**, field **DFHMAH-PROCESSNAME** or a unique value generated by CICS Service Flow Runtime. See “DFHMAH field definitions” on page 81 for further information.
- (**<processtype>**) is equal to value passed in the request message header structure, **DFHMAH**, field **DFHMAH-PROCESSTYPE**. See “DFHMAH field definitions” on page 81 for further information.
- (**<transid>**) is equal to Navigation Manager transaction ID **(CMAM**).
- (**<program>**) > is equal to Navigation Manager program name (**DFHMAMGR**). The Navigation Manager runs as the BTS root activity (DFHROOT).

The NOCHECK processing option indicates that no record is to 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 write to the repository and its associated logging. See the *CICS Business Transaction Services* manual for further information on the **DEFINE PROCESS** command and options.

b. Writes application request data to the process input data-container **ADAPTER.INPUT**. The Adapter navigator program activity reads the input data-container **ADAPTER.INPUT** to retrieve the application request data that is used in Adapter service processing.

c. Writes state information to the process data-container **ADAPTER.PROCESS**.

   State information includes the request name (**DFHMAH-REQUESTNAME**) as specified in the message header (**DFHMAH**) and the program name and transaction that will process the application request data as defined by request properties, **MP-INITIAL-PROGRAMID** and **MP-INITIAL-TRANSID** fields, respectively. For Adapter services of the aggregate connector type, the program and transaction defined and run **will be an Adapter Navigator**. The Adapter Navigator will perform the request processing (i.e., manage the flow navigation of the Adapter service and run the server adapters required to process the business request).

d. Runs the BTS process either in synchronous or asynchronous mode as defined by request property, **MP-REQUESTTYPE**.

   Note: If you are using a synchronous interface (i.e., ECI, EXI or DPL) to invoke server runtime processing, and if the service requestor expects a response, your Adapter service request must be run in synchronous mode. Asynchronous mode processing assumes a WebSphere MQ interface.

   Synchronous BTS RUN command (MP-REQUESTTYPE = 1 or 2)

```
EXEC CICS RUN ACQPROCESS
              SYNCHRONOUS
                    RESP (CICS-RESP)
                   RESP2 (CICS-RESP2)
END-EXEC.
```

Asynchronous BTS RUN command (MP-REQUESTTYPE = 0)

```
EXEC CICS RUN ACQPROCESS
          ASYNCHRONOUS
                   RESP (CICS-RESP)
                  RESP2 (CICS-RESP2)
END-EXEC.
```

DPL Stub program request processing for an Service Flow Runtime service of an aggregate connector type

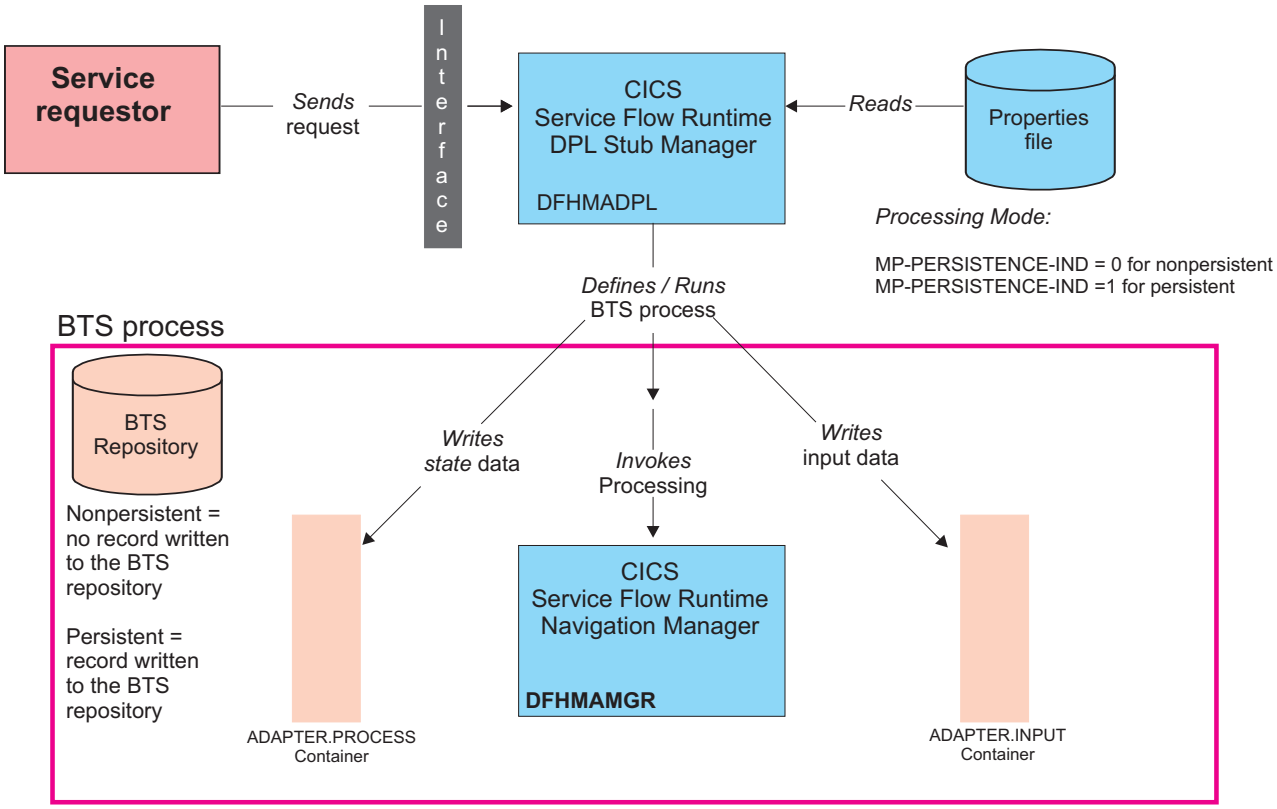


Figure 30. DPL Stub program request processing — aggregate connector type

4. Once invoked, the Navigation Manager (DFHMAMGR) performs the following processing steps:
   a. Reads the adapter state information from process container **ADAPTER.PROCESS**.
   b. Defines the Adapter Navigator BTS activity that will process the application request using the BTS command **EXEC CICS DEFINE ACTIVITY**:

```
EXEC CICS DEFINE ACTIVITY  (ADC-NAV-ACTIVITY)
                    EVENT  (<completionevent>)
                  TRANSID  (ADC-NAV-TRANSACTION)
                  PROGRAM  (ADC-NAV-PROGRAM-ID)
               ACTIVITYID  (ADC-NAV-ACTIVITYID)
                     RESP  (CICS-RESP)
                    RESP2  (CICS-RESP2)
END-EXEC.
```

   where:

- **ADC-NAV-ACTIVITY** is equal to the request name value passed and stored in the process container **ADAPTER.PROCESS**.
- **ADC-NAV-TRANSACTION** is equal to the initial transaction ID as indicated on the request properties, field **MP-INITIAL-TRANSID**, and stored in process container **ADAPTER.PROCESS**
- **ADC-NAV-PROGRAM-ID** is equal to the initial program name as indicated on the request properties, field **MP-INITIAL-PROGRAMID**, and also stored in process container **ADAPTER.PROCESS**.

c. Writes adapter state information to container **ADAPTER.PROCESS**. Additional state information includes the ACTIVITYID, CICS applid where the Navigation Manager (DFHMAMGR) is running, EIBTASKN of the running Navigation Manager, etc.

Note: For Adapter services of the *aggregate connector* type, the program and transaction invoked **will be an Adapter Navigator** . The Adapter Navigator will perform the request processing (i.e., manage the flow navigation of the Adapter service and run the server adapters required to process the business request).

d. Runs the Adapter Navigator BTS activity as defined by the Adapter service, either in synchronous or asynchronous mode, as defined by request property, **MP-REQUESTTYPE**.

Synchronous BTS RUN command (MP-REQUESTTYPE = 1 or 2)

```
EXEC CICS RUN ACTIVITY (ADC-NAV-ACTIVITY)
          SYNCHRONOUS
                 RESP (CICS-RESP)
                RESP2 (CICS-RESP2)
END-EXEC.
```

Asynchronous BTS RUN command (MP-REQUESTTYPE = 0)

```
EXEC CICS RUN ACTIVITY (ADC-NAV-ACTIVITY)
          ASYNCHRONOUS
                 RESP (CICS-RESP)
                RESP2 (CICS-RESP2)
END-EXEC.
```

Navigation Manager request processing of an Service Flow Runtime service of an aggregate connector type

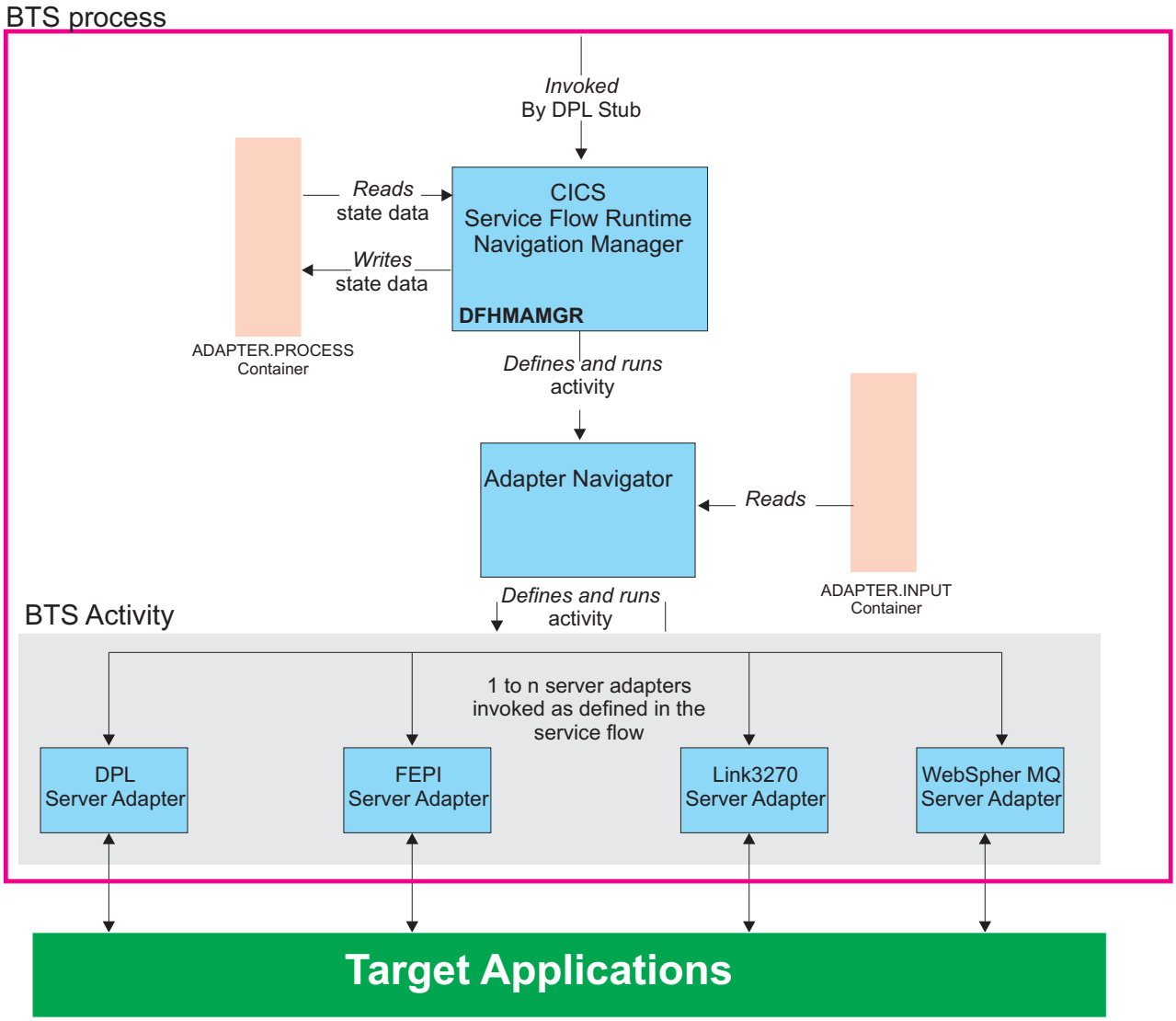


Figure 31. Navigation Manager program request processing — aggregate connector type

As a result of the Navigation Manager (DFHMAMGR) running the Adapter Navigator activity, the DPL, WebSphere MQ, FEPI and/or Link3270 server adapter will perform business request processing (as modelled in Service Flow Modeler) in order to complete the request made by the service requestor.

For example, the generated and deployed Adapter service (an Adapter Navigator and 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 an WebSphere MQ queue.

3. Invoke a server adapter to interact with one or more CICS transactions via 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 via DPL that executes complex rules such as logic or complex input and output. A custom program can be developed to augment CICS Service Flow Runtime.
5. Return application reply data to the Navigation Manager (DFHMAMGR). It could contain the final result.

Each business request issued by the service requestor results in a different BTS process instance to fulfill the appropriate Adapter service as defined by the business request name.

Each process instance consists of a Navigation Manager (DFHMAMGR) activity, an Adapter Navigator activity and those server adapter activities that are needed to support that Adapter service.

Reply processing patterns for Aggregate connector — persistent / nonpersistent

Upon each server adapter completing normally, any application response and reply data is written to a server adapter output data container and control is returned to the Adapter Navigator.

- The DPL and WebSphere MQ output container name is **COMMAND.OUTPUT**
- The FEPI output container name is **FEPI.OUTPUT**.
- The Link3270 output container name is **LINK3270.OUTPUT**.

DPL and WebSphere MQ server adapter status information is written to a status container **COMMAND.STATUS**.

FEPI and Link3270 server adapter status information is written to a status container **NAVIGATOR.STATUS**.

The Adapter Navigator will perform 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:

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

   The value of variable **NSC-CHILD-ACTIVITY** of the ACTIVITY option is equal to the server adapter program name run.
2. Reads the adapter state information from process container **ADAPTER.PROCESS**
3. Reads the appropriate server adapter status information from either container **NAVIGATOR.STATUS** or **COMMAND.STATUS** depending upon server adapter type.

4. Reads the appropriate server adapter output container, if normal completion, from one of the following containers depending, again, upon the server adapter type.
   - **COMMAND.OUTPUT**
   - **FEPI.OUTPUT**
   - **LINK3270.OUTPUT**

The Adapter Navigator determines if normal processing of the Adapter service is complete. If so, the Adapter Navigator will perform the following reply processing steps:

1. Writes output data container containing the application response/reply data to it's output container **ADAPTER.OUTPUT**.
2. Writes adapter service completion status and state information to it's status container **ADAPTER.STATUS**
3. Issues an EXEC CICS RETURN ENDACTIVITY command returning control to the Navigation Manager (DFHMAMGR) root activity thus completing Adapter service processing.

Once all server adapters that constitute the Adapter service have completed processing and the Adapter Navigator activity has ended normally, a BTS completion event is fired and control is returned to the Navigation Manager (DFHMAMGR).

The Navigation Manager will perform the following reply processing steps:

1. Performs a check on the Adapter Navigator activity completion status by issuing the **BTS 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 state information from process container **ADAPTER.PROCESS**.
3. Reads the Adapter service status information from container **ADAPTER.STATUS**.
4. Reads the Adapter service output container, if normal completion, from container **ADAPTER.OUTPUT**.
5. Writes the root activity output data container containing the application response and reply data. The output container name is also **ADAPTER.OUTPUT**.
6. Writes adapter completion status and state information to process container **ADAPTER.PROCESS**.
7. Optionally issues MQPUT if BTS process was executed in asynchronous mode and the client application interface was WebSphere MQ-CICS bridge. Otherwise, the Navigation Manager (DFHMAMGR) issues an **EXEC CICS RETURN ENDACTIVITY** command returning control to the DPL Stub program (DFHMADPL) and completing the process.

Navigator Manager reply processing for an Adapter service of the aggregate connector type

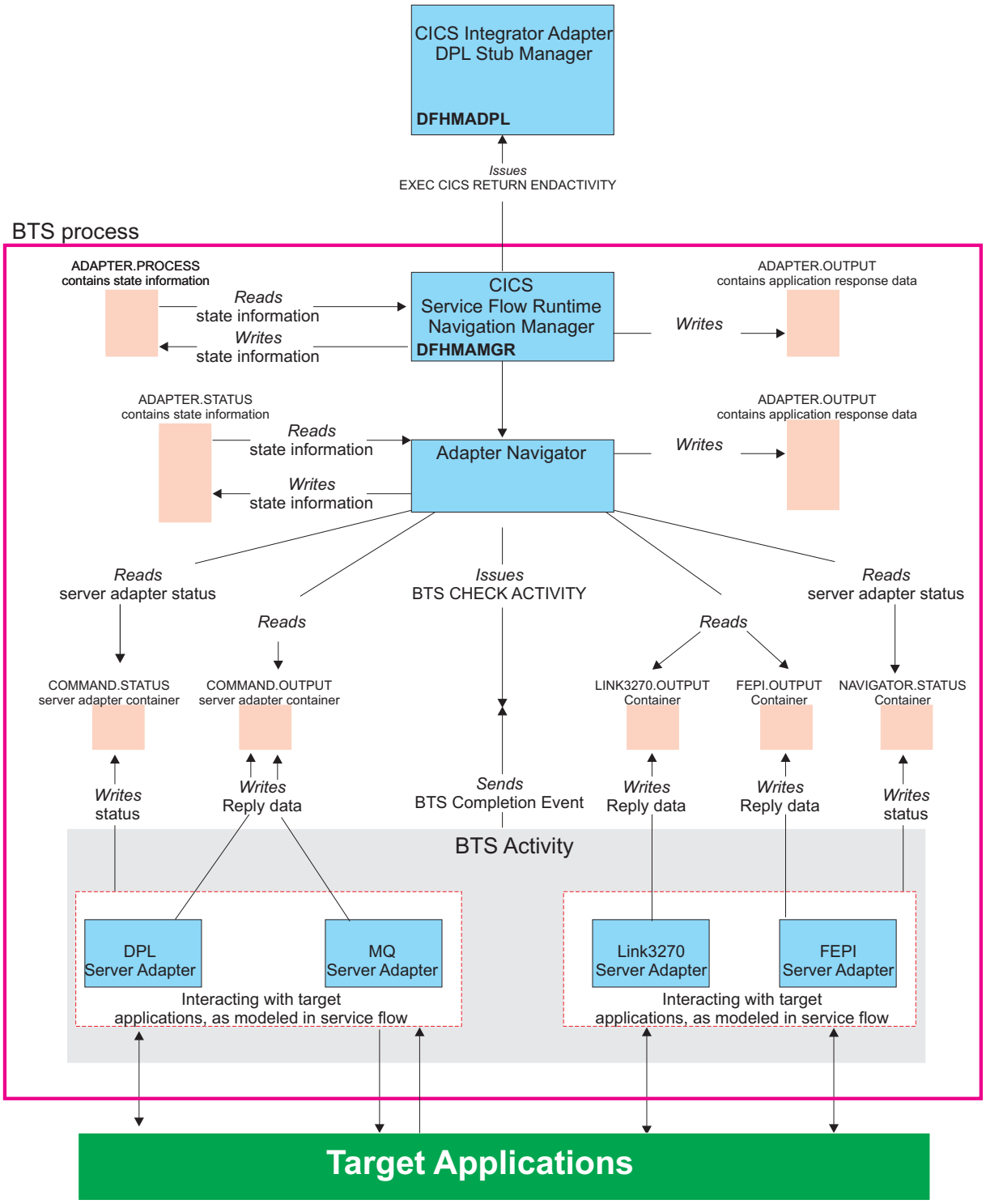


Figure 32. CICS Service Flow Runtime Navigation Manager program reply processing — aggregate connector type

Once the Navigation Manager (DFHMAMGR) has completed processing and the BTS process has ended normally, a BTS completion event is fired and control is returned to the DPL Stub program (DFHMADPL). The DPL stub will perform the following processing steps:

1. Performs a check on the process completion status by issuing the BTS **CHECK ACQPROCESS** command
2. Reads the state information from process container **ADAPTER.PROCESS**.
3. Reads the Navigation Manager output data container **ADAPTER.OUTPUT**.
4. Builds XML or fixed format reply message to include the header structure (DFHMAH) and any output data container application response/reply data.

   Note: The DPL Stub program (DFHMADPL) can call a user-defined program to convert the COBOL fixed format outbound application reply structure to XML application reply tag data. The application program name is determined by the Adapter service.

   The DPL Stub program will then call the COBOL to XML Converter program (DFHMAXMO) to convert the COBOL fixed format header structure (DFHMAH) to XML header tag data to format the XML reply message. See “Request properties” on page 274 for further information.

   See “XML request and response processing” on page 182 and “XML message formats for non-passthrough” on page 322 for further information regarding XML message processing.

   Note: The Navigation Manager program (DFHMAMGR) also performs XML reply message processing as described above when the Adapter service is run in asynchronous mode and the client application interface was WebSphere MQ-CICS bridge.
5. Moves outbound reply message to **DFHCOMMAREA** and issues an **EXEC CICS RETURN** command returning control to the service requestor.

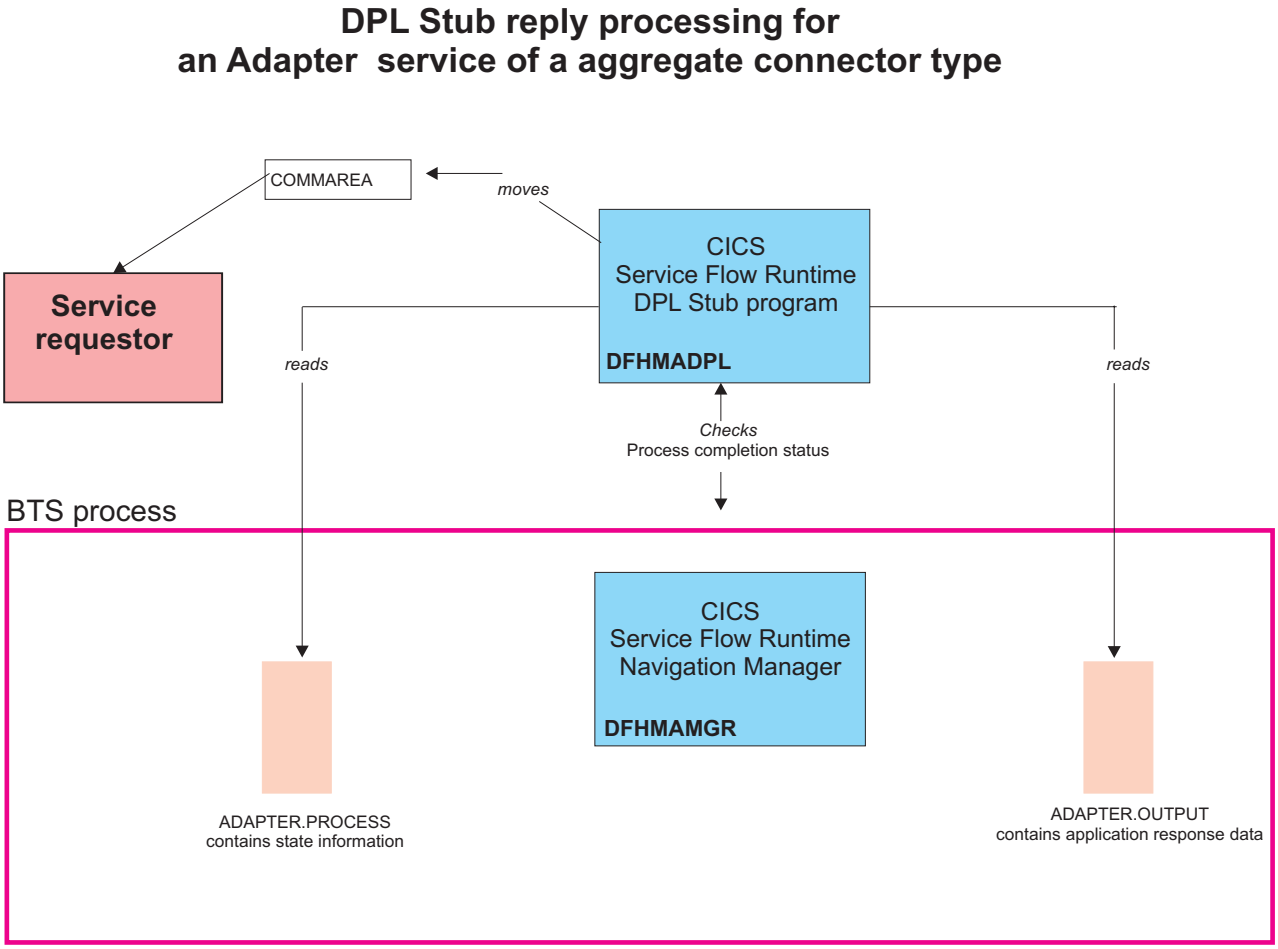


Figure 33. CICS Service Flow Runtime DPL Stub program reply processing — aggregate connector type

Passthrough processing

Passthrough processing refers to the programmatic functions coded in, and managed by the service requester in order to fulfill a business transaction in the CICS Service Flow Runtime.

Passthrough processing does not require developers to model an Adapter service to integrate with CICS target applications, and therefore is sometimes referred to as the non-tooled access pattern. It is supported for CICS Link3270 bridge processing only.

In passthrough processing, the request and response model is based on a *one-screen-at-a-time* conversation between the service requester and the target CICS application. The service requester sends a request message to the runtime environment, in which the application data portion of the message contains a BMS Application Data Structure (ADS) to satisfy:

- A **BMS RECEIVE MAP** command
- A 3270 data stream to either satisfy a **BMS RECEIVE** or **CONVERSE** command
- Initiation of a CICS target transaction using the CICS Link3270 bridge mechanism.

CICS Service Flow Runtime recognizes the request as being a request for passthrough processing and handles it appropriately. In a scenario that uses passthrough processing, the service requester would be designed to handle the passthrough response. The passthrough response could contain multiple Link3270 outbound vectors however.

Using passthrough processing, the service requester is responsible for interpreting any or all BMS Application Data Structures (ADS), textual information sent as the result of a BMS SEND TEXT, CONVERSE or SEND command and any 3270 data streams sent as the result of a SEND or CONVERSE command and based on its interpretation takes appropriate action in the language of the service requester, JAVA for example, to satisfy the command issued by the target CICS application transaction.

The following characteristics apply to the processing of a passthrough request:

- Passthrough request processing runs in synchronous mode only.
- Passthrough processing is supported on Link3270 server adapters only.
- Passthrough processing is always run using the BTS NOCHECK option.

  See“Server runtime processing and the BTS NOCHECK option” on page 114 for information on the BTS NOCHECK option.
- Passthrough request messages must include the message header (**DFHMAH**), the passthrough message header **(DFHMAH2**), the passthrough screen header (**CIA-SCREEN-HEADER**) and the passthrough map header (**CIA-MAP-HEADER**).

  See “Request message headers” on page 80 for further information.
- Passthrough request messages can include only one **CIA-SCREEN-HEADER** structure and only one **CIA-MAP-HEADER** structure with the appropriate inbound Link3270 vector data (ADS, 3270 data stream) necessary to satisfy a RECEIVE, RECEIVE MAP or CONVERSE command issued by the target CICS application transaction.
- Passthrough processing uses a different DPL Stub program (DFHMADPP) than processing of other deployment types.
- Passthrough processing using XML messages uses the load module DFHMAXPI to parse the inbound XML application request data and build the outbound XML application reply data.

See the *CICS External Interfaces Guide Version 2 Release 2* or higher for further information regarding the Link3270 bridge mechanism and associated processing.

Processing patterns for a passthrough request

The following information describes the processing that is associated with a passthrough request.

1. A service requester invokes the runtime environment when it sends a correctly formatted passthrough request message using any of the supported interfaces.

   The request message contains the name of the DPL passthrough stub program (DFHMADPP), which initiates server runtime processing.

   The passthrough request message must include the appropriate header structures and must indicate the following in the header structure (DFHMAH) of the request message:

   - The **DFHMAH-UOWCONTROL** field must be set to + 3 (**DFHMAH-PASSTHROUGH**) to indicate passthrough processing mode.
   - The **DFHMAH-VERSION** field must be set to +2 (**DFHMAH-VERSION-2**).

- The **DFHMAH-FORMAT** field must be set to MAH2 (**DFHMAH-PASSTHROUGH-HDR**) to indicate the DFHMAH2 header structure follows.

  If the request message is in XML format, the DPL Stub program will call the XML Header to COBOL Converter program (DFHMAXMI) to parse the XML header tag data and return in a COBOL fixed format (DFHMAH).

  The DPL Passthrough Stub program (DFHMADPP) will then call the ADS to XML Converter program (DFHMAXPI) to parse the XML application request tag data and return in a COBOL fixed format structure. The inbound application request tag data in passthrough mode must also include pertinent screen header (**CIA-SCREEN-HEADER**) and map header (**CIA-MAP-HEADER**) tag data.

  See "XML request and response processing for passthrough" on page 188 and "XML message formats for passthrough requests" on page 340 for further information.

2. The DPL passthrough stub program (DFHMADPP), receives the request through one of the supported interfaces and performs the following processing steps:

   Note: The DPL Passthrough Stub program **(DFHMADPP) does NOT read** the Properties file (DFHMAMPF) as in the single connector and aggregate connector patterns. The request name (**DFHMAH-REQUESTNAME**) still must be passed in as part of the request message in the message header structure (DFHMAH), however. See "Request message headers" on page 80 for further information.

   a. Defines the BTS process via the BTS command **EXEC CICS DEFINE PROCESS**:

   ```
   EXEC CICS DEFINE PROCESS (<processname>)
                PROCESSTYPE (<processtype>)
                    TRANSID (<transid>)
                   PROGRAM  (<program>)
                  NOCHECK
                       RESP (CICS-RESP)
                      RESP2 (CICS-RESP2)
   END-EXEC.
   ```

   where:

   - (*<processname>*) is equal to value passed in the request message header structure, DFHMAH, field **DFHMAH-PROCESSNAME** or a unique value generated by CICS Service Flow Runtime. See "Request message headers" on page 80 for further information.
   - (*<processtype>*) is equal to value passed in the request message header structure, DFHMAH, field **DFHMAH-PROCESSTYPE**. See "Request message headers" on page 80 for further information.
   - (*<transid>*) is equal to the Navigation Manager transaction ID (CMAM).
   - (*<program>*) > is equal to the Navigation Manager program name (DFHMAMGR). The Navigation Manager runs as the BTS root activity (DFHROOT).

   The NOCHECK processing option indicates that no record is to 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 write to the repository and its associated logging. See *CICS Business Transaction Services* manual for further information on the DEFINE PROCESS command and options.

   b. Writes application request data to the process input data-container **ADAPTER.INPUT**.

   c. Writes state information to the process data-container **ADAPTER.PROCESS**.

State information includes the request name (**DFHMAH-REQUESTNAME**) as specified in the message header (DFHMAH) and the processing mode (**DFHMAH-UOWCONTROL**).

d. Runs the BTS passthrough process in synchronous mode.

```
EXEC CICS RUN ACQPROCESS
            SYNCHRONOUS
                    RESP (CICS-RESP)
                   RESP2 (CICS-RESP2)
END-EXEC.
```

3. Once invoked, the Navigation Manager (DFHMAMGR) will perform the following processing steps:

   a. Reads the state information from process container **ADAPTER.PROCESS**.

   b. Defines the BTS Link3270 Passthrough Manager activity that will process the application request via the BTS command; EXEC CICS DEFINE ACTIVITY:

```
EXEC CICS DEFINE ACTIVITY  (ADC-NAV-ACTIVITY)
                     EVENT  (<completionevent>)
                   TRANSID  (ADC-NAV-TRANSACTION)
                   PROGRAM  (ADC-NAV-PROGRAM-ID)
                ACTIVITYID  (ADC-NAV-ACTIVITYID)
                      RESP  (CICS-RESP)
                     RESP2  (CICS-RESP2)
END-EXEC.
```

   where:

   - **ADC-NAV-ACTIVITY** is equal to the request name value passed and stored in the process container **ADAPTER.PROCESS**.
   - **ADC-NAV-TRANSACTION** is equal to Link3270 Passthrough Manager transaction ID (CMAL)
   - **ADC-NAV-PROGRAM-ID** is equal to Link3270 Passthrough Manager program name (DFHMALPT).

   c. Writes adapter state information to container **ADAPTER.PROCESS**. Additional state information includes the ACTIVITYID, CICS applid where the Navigation Manager (DFHMAMGR) is running, EIBTASKN of the running Navigation Manager, etc.

   Note: For passthrough processing, the program and transaction run **will not** be an Adapter Navigator or a DPL, WebSphere MQ, FEPI or Link3270 server adapter but rather the Link3270 Passthrough Manager load module (DFHMALPT) as there is no modelled Adapter service.

   d. Runs the BTS Link3270 Passthrough Manager activity in synchronous mode.

```
EXEC CICS RUN ACTIVITY (ADC-NAV-ACTIVITY)
          SYNCHRONOUS
                  RESP (CICS-RESP)
                 RESP2 (CICS-RESP2)
END-EXEC.
```

DPL Stub program request processing for a passthrough request

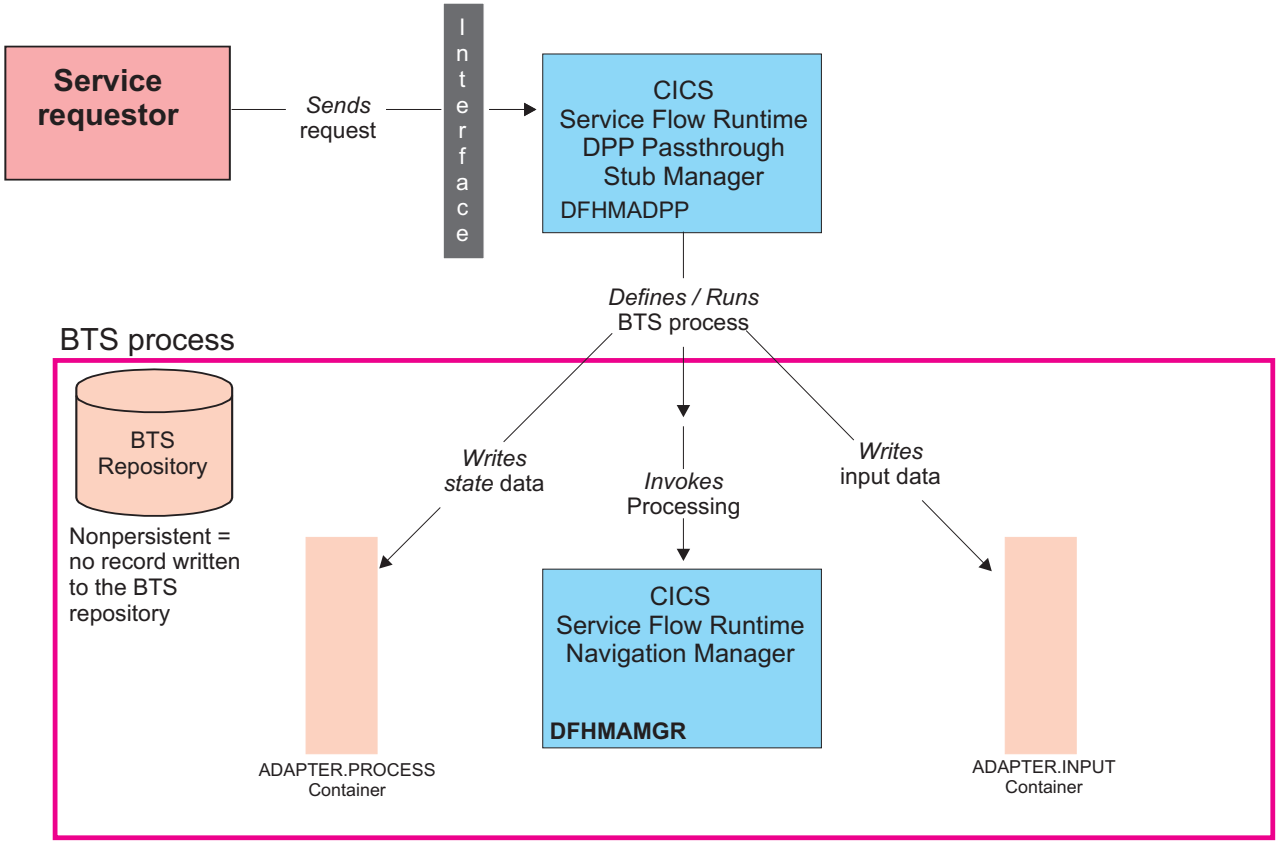


Figure 34. DPL Stub program request processing — Passthrough

The BTS Link3270 Passthrough Manager activity will perform Link3270 passthrough processing in order to complete the request made by the service requestor.

The Link3270 Passthrough Manager (DFHMALPT) performs the following operations:

a. Calls the Link3270 Maintenance program (DFHMALNM).

   The Link3270 Maintenance program performs initialization and exit processing for passthrough processing. It calls the Link3270 Facility State Management program (DFHMALTS), if necessary, to retrieve, store and/or delete Link3270 facility business state information.

b. Calls the Link3270 Vector Processor (DFHMAVCP).

   The Link3270 Vector Processor program sends vectors to and receives vectors from any CICS target application that uses BMS commands (with some restrictions) where a single send and receive structure inclusive of the Link3270 bridge header (BRIH), the inbound/outbound vector header and appropriate inbound/outbound vector is not greater than 32000 bytes (i.e. the COMMAREA passed to the Link3270 bridge program, DFHL3270, is 32,000 bytes).

   The Link3270 Vector Processor program can also call the Link3270 Vector Log program (DFHMAVCL), if vector logging is on, to log inbound and

outbound Link3270 vector header information and data. See “DFHMAH2 header structure” on page 87 for information on configuring for vector logging and additional passthrough header fields and definitions.

Each passthrough request issued by the service requestor results in a different BTS process instance to fulfill that passthrough request.

Each process instance consists of a Navigation Manager (DFHMAMGR) activity and *the one Link3270 Passthrough Manager (DFHMALPT) activity that is needed to support that passthrough request*. With regards to data-container management, passthrough processing processes as a single connector type.

Passthrough request processing

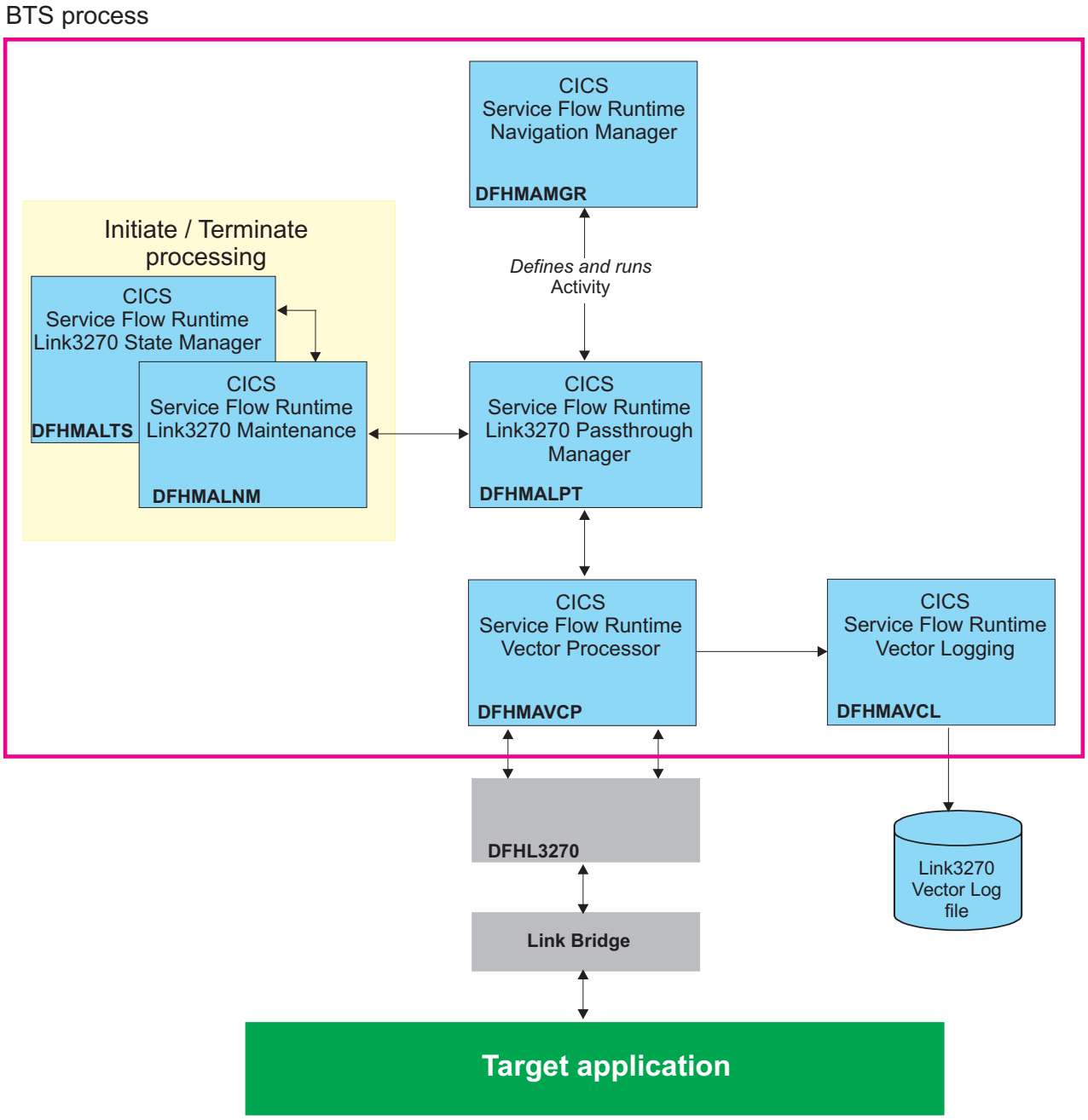


Figure 35. Navigation Manager program request processing — Passthrough

Processing patterns for a passthrough reply

Upon passthrough processing completion, any application response/reply data is written to the Link3270 Passthrough Manager activity's output data container **ADAPTER.OUTPUT**.

The Link3270 Passthrough Manager activity status information is written to the status container **ADAPTER.STATUS**.

Once passthrough processing has completed processing and the Link3270 Passthrough Manager BTS activity has ended normally, a BTS completion event is fired and control is returned to the Navigation Manager (DFHMAMGR).

The Navigation Manager will perform the following reply processing steps:

1. Performs a check on the Link3270 Passthrough Manager activity completion status by issuing the BTS 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 adapter state information from process container **ADAPTER.PROCESS**.
3. Reads the Link3270 Passthrough Manager activity status information from container **ADAPTER.STATUS**.
4. Reads the Link3270 Passthrough Manager activity output container, if normal completion, from container **ADAPTER.OUTPUT**.
5. Writes the root activity output data container containing the application response/reply data. The output container name is **ADAPTER.OUTPUT**.
6. Writes adapter completion status and state information to process container **ADAPTER.PROCESS**.
7. Issues an EXEC CICS RETURN ENDACTIVITY command returning control to the DPL Passthrough Stub program (DFHMADPP) and completing the process.

Once the Navigation Manager (DFHMAMGR) has completed processing and the BTS process has ended normally, a BTS completion event is fired and control is returned to the DPL Passthrough Stub program (DFHMADPP). The DPL stub will perform the following processing steps:

1. Performs a check on the process completion status by issuing the BTS CHECK ACQPROCESS command.
2. Reads the state information from process container **ADAPTER.PROCESS**.
3. Reads the Navigation Manager (DFHMAMGR) output data container **ADAPTER.OUTPUT**.
4. Builds XML or fixed format reply message to include the header structure (**DFHMAH**) and any output data container application response/reply data.

   Note: The DPL Passthrough Stub program (DFHMADPP) calls the ADS to XML Converter program (DFHMAXPI) to convert the COBOL fixed format outbound application reply structure to XML application reply tag data. The outbound application reply structure in passthrough mode includes the screen header (**CIA-SCREEN-HEADER**) and map header (**CIA-MAP-HEADER**) structures.

   The DPL Passthrough Stub program will then call the COBOL to XML Converter program (DFHMAXMO) to convert the COBOL fixed format header structures (DFHMAH and DFHMAH2) to XML header tag data to format the XML reply message.

See “XML request and response processing for passthrough” on page 188 and “XML message formats for passthrough requests” on page 340 for further information.

5. Moves outbound reply message to **DFHCOMMAREA** and issues an EXEC CICS RETURN command returning control to the service requestor.

Invocation of DPL in a FEPI or Link3270 service flow

If the FEPI or Link3270 bridge server adapter uses an inline DPL command, then the run time behavior for the DPL command differs somewhat 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 will contain an embedded **EXEC CICS LINK** command for the DPL command, instead of the generated statements that define and perform a new CICS Service Flow Runtime BTS activity to execute the link.

In Figure 36, *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.

Note: Refer to the Service Flow Modeler help in the *WebSphere Developer for System z* information center for information on the property files.

```
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 36. Invocation of DPL in FEPI service flow

The **EXEC CICS LINK** for the inline DPL command will execute under the same Business Transactions Services (BTS) activity / container management as the FEPI or Link3270 service flow itself. Therefore, the FEPI or Link3270 service flow will execute 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 CICS Service Flow Runtime / BTS activity as is done for DPL commands in a regular (that is, non-FEPI or non-Link3270) service flow.

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

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

DPL server adapter processing

When the DPL server adapter is invoked during request processing, the details of the distributed programming link are passed in BTS data-containers from the adapter navigator or navigation manager to the DPL server adapter.

The adapter performs the link to the CICS application in another CICS region, passing a COMMAREA. When the application returns a response, the server adapter passes the data back to the adapter navigator or navigation manager using BTS data-containers.

The data-containers that are passed to the DPL server adapter store the data that is specified by Service Flow Modeler when the link is modeled. The data for the server adapter is stored in the COMMAND.INPUT data-container. When the server adapter receives a response from the application, it is stored in the COMMAND.OUTPUT data-container and the state information of the DPL adapter is stored in the COMMAND.STATUS data-container.

If you have deployed an adapter service from WebSphere Developer for System z version 7, only one DPL server adapter is generated to handle all of the distributed programming links. This server adapter is called DFHMASDP.

DFHMASDP server adapter processing

The DFHMASDP server adapter runs under the CMAS transaction. The data-containers that are passed to DFHMASDP store the data that is specified by Service Flow Modeler when the link is modeled. The data for DFHMASDP is stored in the COMMAND.INPUT data-container. The information regarding the DPL link itself is stored in the DPL.DATA data-container. When the server adapter receives a response from the application, it is stored in the COMMAND.OUTPUT data-container and the state information of the DPL adapter is stored in the COMMAND.STATUS data-container.

If the target application runs in a different region than the CICS Service Flow Runtime, then the linked-to application program runs under CSMI, the default mirror transaction ID. You can optionally specify a different transaction in Service Flow Modeler. 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, then 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 resources for the transaction
2. Grant access authority in DB2 and RACF as appropriate.

Any DPL server adapter that runs a transaction that makes DB2 calls requires a DB2Trans or DB2Entry resource definition for the transaction. The difference is that the DFHMASDP server adapter has the DB2 access occurring in the transaction under which the invoked target application runs, whereas previously the DB2 access occurred in the transaction under which the adapter service runs.

FEPI server adapter processing

The processing descriptions contained in the following section pertain to Adapter services that include FEPI server adapters.

CICS Service Flow Runtime FEPI file processing

The CICS Service Flow Runtime SLU Connection file (DFHMACOF) is mapped by copybook DFHMARCO in the CICS Service Flow Runtime product library .SCIZMAC. It has an associated alternate file, (DFHMAC1F). The files are used to track the state of connections and conversations and ownership of conversations.

The CICS Service Flow Runtime Target Interaction file (DFHMATIF), is mapped by copybook DFHMARTI in the CICS Service Flow Runtime product library .SCIZMAC. Normally, this file is used to store the last buffer (max. 3600 bytes) sent or received from and to the defined target applid or application transaction as modeled in the service flow. The buffer is used in subsequent FEPI processing when active conversations are acquired (FEPI ALLOCATE PASSCONVID), to determine the processing state of the conversation.

- ***Force, Release, Hold*** indicate the CICS Service Flow Runtime SLU Connection file record must be deleted for the defined connection (i.e., the defined target and node that was in use). There is no buffer written CICS Service Flow Runtime Target Interaction file in this case.
- ***Pass*** indicates the CICS Service Flow Runtime SLU Connection file record for the connection and conversation in-use should not denote any ownership (i.e., SC-LU-OWNER = spaces). The SLU Connection file record is rewritten indicating no ownership of the existing conversation. The conversation is available for use in another task. The Target Interaction file is written with the last buffer sent or received.
- ***Leave assigned*** indicates the CICS Service Flow Runtime SLU Connection file record for the connection and conversation that is in-use should be left assigned or indicate ownership (i.e., SC-LU-OWNER = Userid).

  If your site is implementing *LU owner assignment for non-unique UserIDs*, CICS Service Flow Runtime appends a unique tag to the LU owner assignment on the CICS Service Flow Runtime SLU Connection alternate (DFHMAC1F) file key. This unique tag is comprised of the CICS Applid and EIBTASKN of the CICS Service Flow Runtime DPL Stub program (DFHMADPL). See “Run time processing of non-unique UserIDs using FEPI adapters” on page 159 for more information on how CICS Service Flow Runtime processing works with non-unique UserIDs.

  Note: Userid is the signed-on Userid to the local CICS region as determined by an EXEC CICS ASSIGN command, see the *CICS Application*

Programming Reference.

The conversation is available for subsequent use by the owner of the conversation in another task. The CICS Service Flow Runtime Target Interaction file is written with the last buffer sent or received.

If you use LU assignment processing for non-unique UserIDs and if you have submitted a request to run in synchronous mode, the abnormal termination of that request can result in LUs remaining assigned. The LUs that were assigned will have their sessions (FEPI conversations) terminated and assignment deleted before the CICS Service Flow Runtime / BTS process is complete, successfully executed or not. If the request terminates abnormally in asynchronous mode and you are using LU assignment processing for non-unique UserIDs, the LUs are left for compensation.

Also, as a normal end of day processing strategy, the service requestor could invoke a modeled flow as a process to locate, logoff and release any assigned LUs.

FEPI adapter LU assignment processing for non-unique UserIDs

A non-unique UserID is one that is not associated to a single owner, but instead can be used by more than one user.

LU assignment for non-unique UserIDs provides the following processing:

- LUs within the same FEPI pool are assigned uniquely to a non-unique user UserID and left assigned to that UserID during invocation of potentially multiple FEPI server adapters within the scope of a single executing business request.
- That multiple of these business requests can be executing concurrently for the same non-unique UserID. The processing of each of these concurrent business requests would and must be performed in separate and distinct CICS Service Flow Runtime / BTS processes.
- The provision for LUs to remain assigned, (signed-on and in-session for a specific unique userId) across multiple invocations of subsequent processes that may require usage of that LU. This means the LU could remain assigned to the user UserID on the CICS Service Flow Runtime Connection file after the process is complete.

LUs are required to establish connections and sessions to CICS regions and applications in modeled flows that contain FEPI Navigators. LU assignment processing for non-unique UserID looks at processing requirements that arise when non-unique UserIDs are used with CICS Service Flow Runtime.

Using LU assignment processing for non-unique UserIDs in your FEPI service flows will determine how the CICS Service Flow Runtime server run-time assigns LUs that are used to establish connections and sessions to CICS regions and applications. See “Run time processing of non-unique UserIDs using FEPI adapters” on page 159 for a description of run time processing associated with LU assignments for non-unique UserIDs .

Implementing LU assignment processing for non-unique UserIDs with FEPI adapters

At build time, the person who models the adapter can enable LU assignment processing for non-unique UserIDs for a FEPI service flow.

For information on how to enable LU assignment processing for non-unique UserIDs, see the section on setting properties in the Service Flow Modeler help.

When you enable LU assignment processing for non-unique UserIDs in your adapter model, you must also provide instructions in your adapter for the release of the assigned LUs. All assigned LUs that were used by all of the FEPI Navigators, and executed as part of a process, must be *released*. Released means that the LU cannot be in a release type of *Assigned*. If you do not release the LU(s), they will remain assigned on the CICS Service Flow Runtime Connection file and will be rendered useless to standard CICS Service Flow Runtime processing. This is true regardless of the processing mode; synchronous processing for inquiry requests; or asynchronous processing for update requests. For information on how set the release type, see information on **MAT_LOGOFFTYPE** in the information on setting properties the Service Flow Modeler help.

Run time processing of non-unique UserIDs using FEPI adapters

LU assignment processing for non-unique UserIDs assumes that UserIDs at your site are not unique.

If you use non-unique UserIDs, you will not be able to take advantage of some CICS Service Flow Runtime processing capabilities. See “Why you should use unique user ids with CICS Service Flow Runtime” on page 178 for information about the benefits of unique UserIDs.

Note:

In a configuration where the user UserID is not unique, CICS Service Flow Runtime run time processing ensures that LUs within the same FEPI pool are *assigned uniquely* to the non-unique UserID. CICS Service Flow Runtime accomplishes this by appending a *unique tag* to the LU owner assignment on the CICS Service Flow Runtime V3.1 SLU Connection alternate (DFHMAC1F) file key. This unique tag is comprised of the CICS Applid and EIBTASKN of the CICS Service Flow Runtime DPL Stub program (DFHMADPL).

The unique tag, which is retrieved in the CICS Service Flow Runtime DPL Stub program (DFHMADPL), is used in conjunction with the user UserID as part of the LU assignment during process execution. The run time processing of an LU assignment for a non-unique UserID using FEPI adapters is as follows:

- In the first FEPI Navigator invoked during the execution of a process, the LU owner assignment is updated on the CICS Service Flow Runtime V3.1 SLU Connection alternate file (DFHMAC1F) key using the unique tag in conjunction with the user UserID and pool name.
- Any subsequent FEPI Navigator executed as part of the same process will use this same unique tag identifier to retrieve the correct assigned, in-session LU.
- The assignment indicator configuration remains in effect for the duration of the business request, which may include the invocation of multiple FEPI navigators.
- The unique tag identifier will allow for concurrent process execution by the same non-unique user UserID, and it will not change during the execution of a single process.

If the process execution terminates abnormally, the service requestor is responsible for handling the condition by one of the following methods:

- initiating a process of a modeled flow to *compensate* for the failed process
- issuing a *cancel* request to the CICS Service Flow Runtime server run-time to release BTS resources.

See “Deciding on whether to cancel or run compensating flow” on page 60 for more information.

In addition to the processing associated with LU assignment for non-unique UserIDs, CICS Service Flow Runtime V3.1 server run-time provides *clean-up* logic in the DPL Stub program. This clean-up logic is invoked when the process terminates abnormally resulting in LUs remaining assigned. See “LU assignment processing for non-unique IDs — synchronous mode” and “LU assignment processing for non-unique IDs — asynchronous mode” on page 162 for a description of how the DPL Stub program implements clean-up logic to remove the LU assignments that remain on the SLU Connection alternate (DFHMAC1F) file key after a process terminates abnormally.

See “CICS Service Flow Runtime FEPI file processing” on page 157 for a description of how CICS Service Flow Runtime uses the CICS Service Flow Runtime SLU Connection file.

LU assignment processing for unique IDs — synchronous mode: When you use unique UserIDs the FEPI LUs are assigned and released as modeled in the Service Flow Modeler. The CICS Service Flow Runtime server run-time makes no attempts to locate and cleanup LUs left assigned upon successful or unsuccessful execution of processes.

Since only the UserID is used for LU assignment and assumed unique in this mode, any and all LUs used and left assigned for the specific UserID can be located and used in the execution of subsequent processes by that UserID.

If FEPI LUs are left assigned in this mode for use in subsequent process execution, the service requestor is responsible to logoff / cleanup LUs for a specific UserID when use of that UserID is no longer required. For example, as a normal end of day processing strategy, the service requestor could invoke a modeled flow as a process to locate, logoff and release any assigned LUs. Failure to incorporate a cleanup strategy for a specific user may not leave LUs available over time to other users — unless the customer configuration is such that a significant number of LUs are available for users requiring access via CICS Service Flow Runtime FEPI processing.

LU assignment processing for non-unique IDs — synchronous mode: If your site uses LU assignment processing for non-unique UserIDs, and if the process terminates abnormally due to a CICS Service Flow Runtime system error or an incorrectly modeled flow, this can result in LUs remaining assigned.

The CICS Service Flow Runtime / BTS process that was initiated to execute that modeled flow will, upon issuing a reply to the service requestor, *complete*.

Under the complete state there will be no CICS Service Flow Runtime / BTS process container information available for subsequent use to locate, use and / or logoff assigned LUs.

For this reason, the LUs that were assigned will have their sessions (FEPI conversations) terminated and assignment deleted before the CICS Service Flow Runtime / BTS process is complete, successfully executed or not.

The CICS Service Flow Runtime DPL Stub program (DFHMADPL), will cleanup / process LUs assigned in this manner before issuing the reply of an executed process to the service requestor.

The DPL Stub program uses the unique tag applied to LU assignments for non-unique UserIDs, in conjunction with the user UserID to:

- locate all LU(s) used in the process.
- terminate LU(s) sessions (FEPI conversations)
- delete the assigned LU from the CICS Service Flow Runtime Connection file record(s)
- initialize CICS Service Flow Runtime Target Interaction file record(s)

The CICS Service Flow Runtime / BTS process will then be completed.

Note: If the cleanup / processing of assigned LUs fails, it will render those LUs useless in normal CICS Service Flow Runtime server run-time processing. There is no way to locate, use or release these LUs in subsequent CICS Service Flow Runtime processes. It is essential that process execution of the modeled flow attempt to logoff and release LUs assigned using this method.

LU assignment processing for unique IDs — asynchronous mode: FEPI LUs are assigned and released as modeled. The CICS Service Flow Runtime server run-time makes no attempts to locate and cleanup the LUs left assigned upon successful or unsuccessful execution of processes.

It is still necessary to *compensate* or *cancel* CICS Service Flow Runtime / BTS processes as the result of unsuccessful process execution. However, in this mode it is not necessary that a flow modeled for the purposes of compensation logoff / cleanup assigned LUs as part of its compensation logic.

Since only the UserID is used for LU assignment and that UserID is assumed to be unique, any and all LUs used and left assigned for the specific UserID can be located and used in the execution of subsequent processes by that UserID.

Also if the only requirement, as determined by the service requestor, is to *cancel* an unsuccessfully executed process, the DPL Stub program (DFHMADPL) issues the BTS CANCEL ACQPROCESS command to end and complete the failed CICS Service Flow Runtime / BTS process.

The DPL Stub program (DFHMADPL) will not terminate LU sessions or remove the assignment on the CICS Service Flow Runtime Connection file record(s) because when only the user UserID is used for LU assignment, the UserID is unique and can be found on any subsequent process execution.

The service requestor may want to *cancel* the failed CICS Service Flow Runtime BTS process but not logoff and release LUs assigned to that UserID.

Subsequent modeled flows can be initiated as processes to locate, use and / or release any assigned LUs as desired.

Note: As is the case in synchronous processing for unique UserIDs, if FEPI LUs are left assigned use in subsequent process execution, the service requestor is responsible to logoff / cleanup LUs for a specific UserID when use of that UserID is no longer required.

For example, as a normal end-of-day processing strategy, the service requestor can invoke a modeled flow as a process to locate, logoff and release any assigned LUs. Failure to incorporate a cleanup strategy for a specific user may leave LUs unavailable to other users over time— unless

the customer configuration is such that a significant number of LUs are available for users requiring access via CICS Service Flow Runtime FEPI processing.

LU assignment processing for non-unique IDs — asynchronous mode: If your site uses LU assignment processing for non-unique UserIDs, and if the process terminates abnormally due to a CICS Service Flow Runtime system error or an incorrectly modeled flow, this can result in LUs remaining assigned. An abnormal termination of a process does not complete/end the CICS Service Flow Runtime /BTS process that was initiated for that modeled flow. This leavesCICS Service Flow Runtime / BTS process container information available from the failed process for subsequent use. This is true regardless of the LU assignment method processing.

These failed CICS Service Flow Runtime / BTS processes must be addressed by the service requestor, again regardless of the method of LU assignment or regardless of whether FEPI is implemented as part of the modeled flow. Since the CICS Service Flow Runtime / BTS process is not complete, BTS resources are still allocated with the state of the CICS Service Flow Runtime / BTS process and process container information.

In order to complete the failed CICS Service Flow Runtime / BTS process and release those BTS resources, the service requestor must perform one of the following actions:

- initiate a process of a modeled compensating flow

  Note: The compensating flow must be run in the same mode as the original flow. This means if the original flow was configured to use non-unique UserID, you need to configure your compensating flow to use the LU assignment processing for non-unique UserIDs.
- issue a cancel request of the failed process to the CICS Service Flow Runtime server run-time to release the BTS resources and complete / end the CICS Service Flow Runtime process.

If the service requestor initiates a process of a modeled compensating flow, the DPL Stub program will:

- ACQUIRE the failed process using the failed process name and process type
- initiate a new CICS Service Flow Runtime / BTS process under which the compensating flow will run
- retrieve (issue GET CONTAINER) the information stored in the containers of the failed process
- issue a BTS CANCEL ACQPROCESS command thus completing the failed CICS Service Flow Runtime / BTS process and releasing BTS resources.
- place that information into the containers associated with the new CICS Service Flow Runtime / BTS process
- RUN the compensating flow process as modeled in the Service Flow Modeler

The unique tag used in the new LU assignment method is included in the container information retrieved from the failed CICS Service Flow Runtime / BTS process and is available to the compensating process.

The unique tag is used to locate all LU(s) used in the failed process.

The LU(s) are used in the compensating process as indicated in the flow model. The LUs ultimately, however, must be released as part of that compensating process.

If the service requestor issues a *cancel* of the failed process, the DPL Stub program will:

- ACQUIRE the failed CICS Service Flow Runtime / BTS process thus gaining access to the container information of that failed CICS Service Flow Runtime / BTS process.

  The unique tag used in the LU assignment for non-unique UserIDs is included in this container information.
- use the unique tag retrieved in conjunction with the user UserID to locate all LU(s) used in the failed process.
- terminate LU(s) sessions (FEPI conversations)
- delete the assigned LU CICS Service Flow Runtime Connection file record(s)
- initialize CICS Service Flow Runtime Target Interaction file

  The CICS Service Flow Runtime Target Interaction file should contain the last screen buffer/data sent/received for the assigned LU.
- issue a BTS CANCEL ACQPROCESS command thus completing the failed CICS Service Flow Runtime / BTS process and releasing BTS resources.

Incorrectly modeled flows that leave LUs assigned in this mode of processing and that complete successfully, may cause those LUs to be rendered useless in normal CICS Service Flow Runtime server run-time processing. There may be no way to locate, use or release these LUs in subsequent CICS Service Flow Runtime processes. cleanup logic in the CICS Service Flow Runtime Navigation Manager program (DFHMAMGR), similar to the logic added to the CICS Service Flow Runtime DPL Stub program (DFHMADPL), attempts to locate and cleanup / process LUs left assigned upon successful process execution. If, however, the cleanup/processing of assigned LUs is not successful in this mode and assignment method, it will render those LUs useless in normal CICS Service Flow Runtime server run-time processing. There is no way to locate, use or release these LUs in subsequent CICS Service Flow Runtime processes. It is essential that process execution of the modeled flow attempt to logoff and release LUs assigned using this method.

Link3270 server adapter processing

The Link3270 server adapter enables a service requester to conduct an interactive BMS request and reply dialog with 3270 application programs that are running in CICS by using the CICS Link3270 bridge mechanism.

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 ADAPTER.INPUT data-container as input to a symbolic map. The Link3270 bridge mechanism satisfies the **RECEIVE MAP** command issued by the target 3270 application.
- Parses the BMS application data structure that is sent by the CICS target application in a **SEND MAP** vector. The target application can set the field focus in the normal way, by placing a `-1` in the appropriate output field length.

- Identifies the transaction, mapset name, map name and its fields, attributes and data.
- Constructs and sends an appropriate **RECEIVE MAP** vector using the BMS application data structure, based on the modeled flow and on simple business logic.
- Handles the next **SEND MAP** vector by parsing, identifying and constructing another **RECEIVE MAP** vector or keystroke, and so on. The Link3270 bridge mechanism passes the symbolic map to the Link3270 server adapter where it can be used for the next transaction, or to format and return the output data-container called ADAPTER.OUTPUT.
- Manages state information for the respective CICS user ID and bridge facility.
- Deletes the Link3270 bridge facility

Target 3270 application programs can use multiple output maps contained in one or more map sets. The map set load modules provide the definition of multiple maps used by the target 3270 application transaction programs. Each map set load module must be defined and available to the router region for Link3270 server adapter processing. This means that any map set used in target 3270 application transaction programs must be in a load library in the DFHRPL DD statement or concatenation of the CICS region in which CICS Service Flow Runtime is installed. Each map set load module must also be defined in that CICS system definition (CSD) file.

As the Link3270 server adapter provides 3270 emulation at the terminal level, it 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. It is the saved application data structure, with the accumulated data, that is used in Link3270 server adapter processing.

Having been presented with the application data structure, the Link3270 server adapter can move business data from the application data structure to any output data container as modelled in the flow. If additional business data is required the Link3270 server adapter must submit additional transactions to the Link3270 bridge mechanism. To do this, input data may need to be supplied for certain fields on the current application data structure. The Link3270 server adapter may obtain this data either from any of it's input data containers or from data collected from previously run target 3270 applications.

Transaction routing processing

Link3270 server adapters support both dynamic transaction routing and transaction routing. The target CICS application transactions can be defined as remote or 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 in the router region and passing a COMMAREA that identifies the transaction to run and the data used by the target CICS application.

Link3270 server adapter processing steps in a transaction routing scenario are as follows:

1. Run terminate processing in current target CICS region (bridge facility allocated)
2. Deallocate currently allocated bridge facility
3. Allocate new bridge facility in the new indicated target CICS region

4. Run initiate processing where (TRANSaction = remote CICS system name/CONNECTION name with REMOTEName = CMAI or TRANSaction = local CICS system name /CONNECTION name with PROGram = DFHMALIN) in new target CICS region.
5. Run target CICS application transaction routed to this new target CICS region.

For further information regarding CICS function request shipping (transaction routing method), see the *CICS Intercommunication Guide*. For programming information about the dynamic transaction routing program, see the *CICS Customization Guide*.

Note: New parameters on the interface allow the dynamic transaction routing program to identify Link3270 bridge requests and obtain the names of the target transaction and the bridge facility token.

For further information about the Link3270 bridge mechanism and its support for these types of intercommunication methods, see the section on "Transaction Routing considerations" and the section on "Using Link3270 bridge load balancing" in the *CICS External Interfaces Guide Version 2 Release 2*.

Configuring the runtime environment to use transaction routing

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

If you want to use dynamic transaction routing or transaction routing for processing request messages using Link3270 server adapter, you must set an AOR routing indicator in the Link3270 server adapter service flow node properties. For information on how to set the routing indicator (MAT_AOR_ROUTING), see the Service Flow Modeler help in the WebSphere Developer for System z information center. If you do not set this indicator, your adapter service might not function correctly.

The router 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 router region when processing using Link3270 server adapter programs, the transaction and program definitions described below are not necessary.

1. Specify a transaction definition in the router 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 will require a definition in the router region.
2. The program DFHMALIN and transaction ID CMAI must be defined in each remote CICS region where your target CICS application transactions run.
3. Define program and transaction definitions for the program DFHMALIN in the router region. The transaction ID should be equal to the CICS system name

(CONNECTION name) of the CICS region, not CMAI. The CMAI transaction definition in the router region is reserved and used for dynamic transaction routing.

4. To perform transaction routing:
   a. Define a transaction in the router 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 router region, specify a transaction definition where the transaction ID is equal to the local CICS system name (i.e., the router 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. The following transaction and program must also be defined in all target CICS application regions:

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

   See Figure 37 on page 167 for an illustration of a configuration using transaction routing.

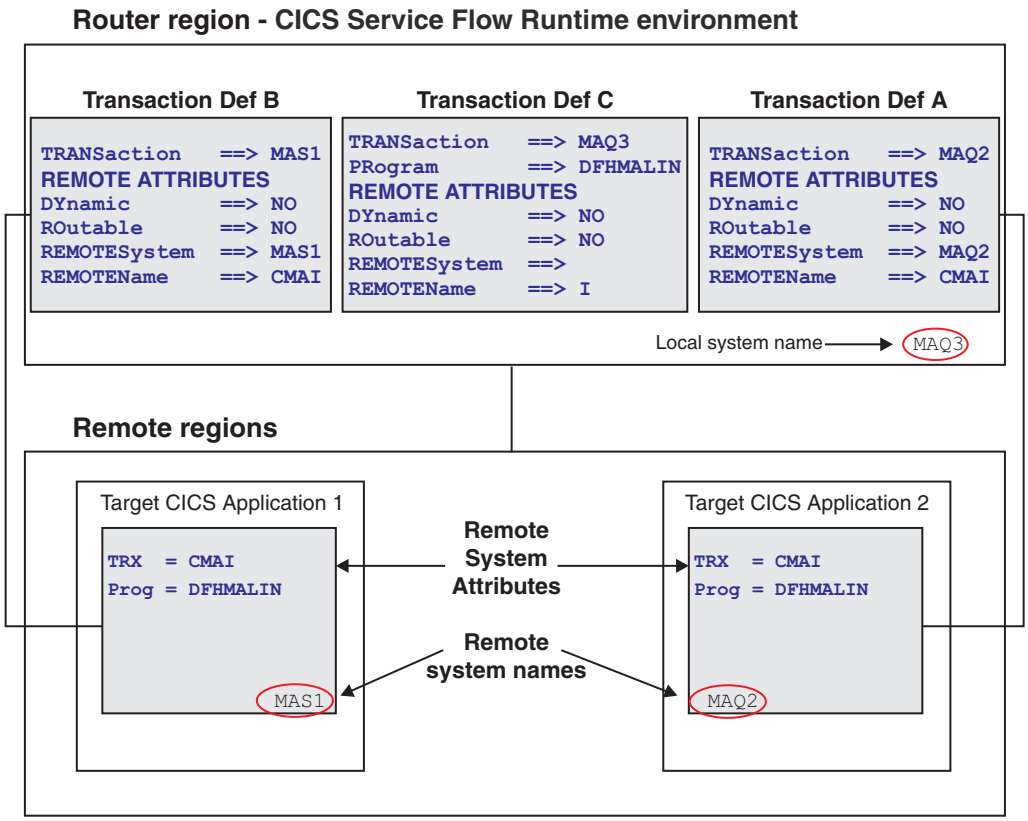


Figure 37. Transaction routing configuration

5. To perform dynamic transaction routing:
   a. Include a transaction definition for the transaction CMAI in the router region and code your dynamic transaction routing program accordingly. This definition is required in addition to the transaction definitions for each target CICS region. In dynamic transaction routing, this definition will be used to perform the initiate processing. The CMAI transaction should be defined with 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.

See Figure 38 on page 168 for an illustration of a configuration using dynamic transaction routing.

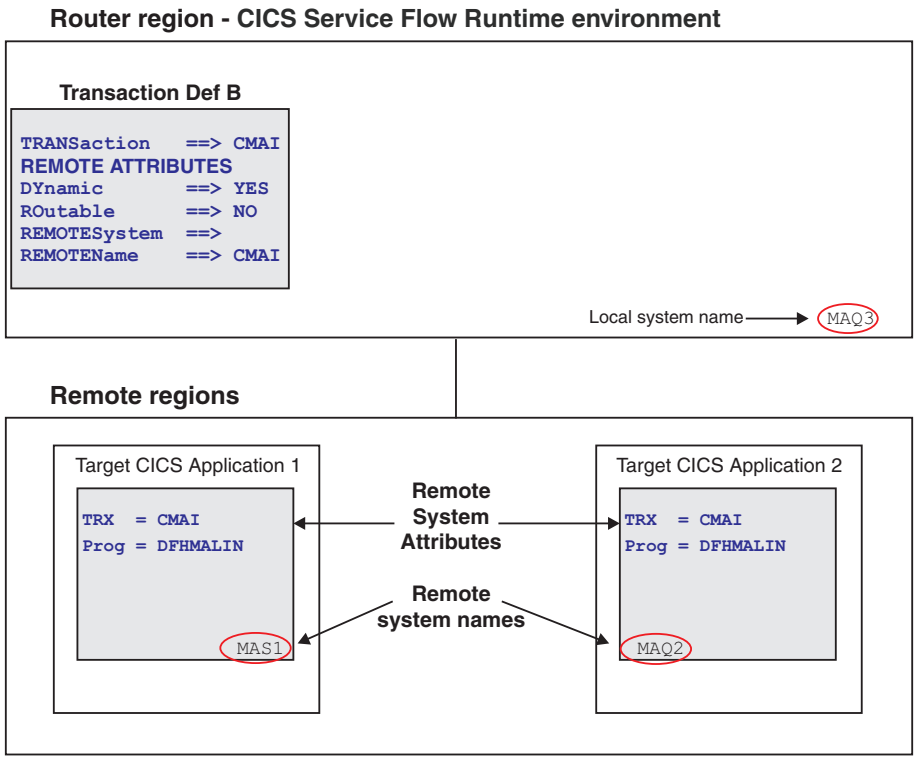


Figure 38. Dynamic transaction routing configuration

Facility state cleanup processing in the CICS Service Flow Runtime

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

CICS Service Flow Runtime includes processing that can clean up facility state information. The runtime environment supports facility state cleanup processing for Link3270 server adapters.

The facility and state information referred to in the following processing descriptions refers to business state data saved for processing sequence flows generated and deployed using the Service Flow Modeler. 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 mechanism.

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

- Facility state cleanup processing — TSQ
- Facility state cleanup processing — VSAM

Facility state cleanup processing — TSQ

If you are running Link3270 Adapter services of the single connector, nonpersistent type, facility state cleanup processing is run 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 and invokes the processing to deallocate the associated Link3270 bridge facilities that CICS has not automatically deleted due to the facility being inactive for the keeptime interval.

Note: See the *CICS External Interfaces Guide* for more information on the keeptime interval processing by CICS.

Processing is as follows:

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

   CICS Service Flow Runtime facility state 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 CICS Service Flow Runtime Link3270 Facility Deallocate Cleanup program (DFHMALFD):
   - A matching TS queue is found and not in use.

     A check is made to see if the CICS Service Flow Runtime facility session state expiration time has been exceeded.
   - The facility session state expiration time has been exceeded

     A check is made to see if the Link3270 bridge facility exists 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. Once 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 30 for information on setting the SI interval.
4. The CICS Service Flow Runtime Link3270 Facility Deallocate Cleanup program de-allocates existing bridge facilities and deletes the associated CICS Service Flow Runtime facility session state data whether that data is stored in TS queue or VSAM data set.

Facility state cleanup processing — VSAM

The CICS Service Flow Runtime Facility Sate Cleanup (VSAM) program (DFHMALFC) is used to manage facility state cleanup processing when you are running Link3270 Adapter services of the following types:

- Aggregate connector, persistent
- Aggregate connector, nonpersistent
- Single connector, persistent

DFHMALFC browses the Link3270 State file and initiates the processing to delete the expired CICS Service Flow Runtime Link3270 facility session state data and invokes the processing to deallocate the associated Link3270 bridge facilities that CICS has not automatically deleted due to the facility being inactive for the keeptime interval. See the *CICS External Interfaces Guide* for more information on the keeptime interval processing by CICS.

The Link3270 State file used by DFHMALFC can vary depending on the build time tool used to generate the sequence flows. DFHMALFC uses the CICS Service Flow Runtime Link3270 State file (DFHMALSF) when supporting flows that were modeled and generated in version 1.1.3 of MQSI Agent for CICS . If the sequence flow was modeled and generated using Service Flow Modeler, DFHMALFC browses the CICS Service Flow Runtime Link3270 file (DFHMAL2F). DFHMALSF provides upward compatibility for customers who choose not to regenerate their Link3270 flows. However, the recommendation is that customers should regenerate their Link3270 flows.

Processing is as follows:

1. DFHMALFC browses Link3270 Facility State file.
2. If the following conditions are true, DFHMALFC calls the CICS Service Flow Runtime Link3270 Facility Deallocate Cleanup program (DFHMALFD):
   - A matching record is found and not in use.

     A check is made to see if the CICS Service Flow Runtime facility session state expiration time has been exceeded.
   - The facility session state expiration time has been exceeded

     A check is made to see if the Link3270 bridge facility exists and is not in a 'RELEASED' state.
3. If the Link3270 bridge facility has been deleted, the record containing CICS Service Flow Runtime facility session state data is deleted, and the browse of the Link3270 State file is resumed. Once 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 30 for information on setting the SI interval.
4. The CICS Service Flow Runtime Link3270 Facility Deallocate Cleanup program de-allocates existing bridge facilities and deletes the associated CICS Service Flow Runtime 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, there are special considerations when using the Link3270 bridge facility. This is because there is a unique Link3270 bridge facility for every CICS region.

If you form all or part of the name of shared temporary storage queues from the terminal id, the same terminal id must be used for each Link3270 bridge facility in a pseudo-conversation 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.

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.

For more information about DFHZATDX, see "Allocating a bridge facility name for a pseudo-conversation when using he Link3270 bridge for transaction routing in the *CICS External Interfaces Guide* and "Writing a program to control autoinstall of terminals" in the *CICS Customization Guide*.

Managing state cleanup for Link3270 server adapters

The Link3270 bridge facility that is allocated in the initial region at the start of the pseudo-conversation is referred to as the primary Link3270 bridge facility. It is important that the application temporary storage queues are not deleted until the completion of the pseudo-conversation, at which time the primary Link3270 bridge facility is deleted. The XFAINTU global user exit runs when the Link3270 bridge facility expires, so you can add functionality to this user exit to ensure that the temporary storage queues are not deleted until the pseudo-conversation is complete.

When the primary Link3270 bridge facility is allocated, CICS SFR writes a temporary storage queue record containing the primary bridge facility token. The terminal id is used as a portion of the queue name. Use the provided sample Figure 39 on page 173 to check if the expired bridge facility is the primary facility. You need to write a program that deletes the temporary storage queues, that is invoked by XFAINTU if it matches the primary facility token with the expired Link3270 bridge facility.

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* should 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.

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.

```
         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
```

Use restrictions for Link3270 bridge mechanism

The Link3270 bridge mechanism has as a prerequisite *CICS Transaction Server for z/OS Version 2 Release 2* or higher. However, the Link3270 bridge mechanism CICS API support varies depending upon the version and release of *CICS Transaction Server for z/OS*.

- See *CICS External Interfaces Guide for CICS Transaction Server for z/OS Version 2 Release 2* for Link3270 bridge restrictions and unsupported EXEC CICS commands in a CICS TS 2.2 environment.
- See *CICS External Interfaces Guide for CICS Transaction Server for z/OS Version 2 Release 3* or higher for Link3270 bridge restrictions and unsupported EXEC CICS commands in a CICS TS 2.3 or higher environment.

Note: CICS Service Flow Runtime only supports the Link3270 bridge mechanism. The START BREXIT bridge mechanism supported in CICS versions prior to *CICS Transaction Server for z/OS Version 2 Release 2* is not supported.

Although CICS Service Flow Runtime has a prerequisite for *CICS Transaction Server for z/OS Version 3 Release 1*, there is no such prerequisite for the target CICS application regions (i.e., the CICS region where your user transactions and target applications are running). If CICS Service Flow Runtime is running user transactions via the Link3270 bridge mechanism and if those user transactions are running in *CICS Transaction Server for z/OS Version 2 Release 2 environment*, CICS Service Flow Runtime and those user transactions are restricted to the *CICS Transaction Server for z/OS Version 2 Release 2* Link3270 bridge mechanism supported CICS APIs.

If your user transactions are running in *CICS Transaction Server for z/OS Version 2 Release 3*, via the Link3270 bridge mechanism and those user transactions are running in a *CICS Transaction Server for z/OS Version 2 Release 3* environment, then CICS Service Flow Runtime and those user transactions are restricted to the *CICS Transaction Server for z/OS Version 2 Release 3* Link3270 bridge mechanism supported CICS APIs.

See the IBM provided sample restrictions table, DFHEIDBR, in the CICS provided sample library (hlq.SDFHSAMP) and the *CICS External Interfaces Guide* for your version(s) and release(s) of *CICS Transaction Server for z/OS* for further information.

The restrictions table, DFHEIDBR, identifies commands which may not be supported when the generated CICS Service Flow Runtime Link3270 server adapter is run using the Link3270 bridge mechanism. This table can be used with an IBM supplied load module scanner utility, DFHEISUP, to analyze your user transactions/target applications. See the *CICS Operations and Utilities Guide for CICS Transaction Server for z/OS Version 2 Release 2* and/or Version 2 Release 3 or higher.

If the scanner shows that these commands are found, then please refer to the section on **Bridging to 3270 transactions** in the appropriate *CICS External Interfaces Guide for CICS Transaction Server for z/OS* for further information. There are other restrictions which may limit the generated Link3270 server adapter functionality when using the Link3270 bridge mechanism, therefore t**his table is a guide rather than a definitive statement**.

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

- User transactions can not issue an EXEC CICS START command to start other local/remote user transactions (local/remote). The target applications should use EXEC CICS RETURN TRANSID() IMMEDIATE, instead. In addition, user transactions that issue an EXEC CICS START command with FROM option for print purposes.
- Additional CICS API commands, not specified in member DFHEIDBR, that are not supported in the user transactions include the following:
  - CONVERSE

    This command is supported in Passthrough mode only.
  - SEND TEXT MAPPED
  - SEND TEXT NOEDIT
  - SEND MAP with ACCUM option

    The SEND MAP with ACCUM option is supported in passthrough mode only. In CICS Service Flow Runtime passthrough processing, each SEND MAP (with ACCUM), vector data area received is returned to the service requestor as a separate map with its own map header, CIA-MAP-HEADER. There is no attempt to merge these vectors as in BMS processing where the output is to a real terminal.
  - SEND of 3270 data streams - supported in passthrough mode only.
  - SEND support is limited to textual information or where a single field is present.

    In both cases, the SEND LENGTH() cannot exceed 132 bytes.
  - SEND TEXT and SEND TEXT with ACCUM, HEADER and TRAILER options are supported in both passthrough and Link3270 server adapter processing.

    Please note that there is no attempt to merge the SEND TEXT vector data areas based upon screen position or justification (JUSTIFY) as in BMS processing where the output is to a real terminal.

    In Link3270 server adapter processing, each SEND TEXT vector data area is accumulated in a text buffer area, each SEND TEXT vector data area is separated by a low-values byte, and returned to the Link3270 server adapter where, if tooled, it can be parsed. (Any HEADER and TRAILER data areas are also stored within the text buffer area.)

    In passthrough processing, each SEND TEXT (no ACCUM) vector data area received in CICS Service Flow Runtime is returned as a separate data area to the service requestor. The SEND TEXT (ACCUM) vector data areas received are accumulated in a buffer as with Link3270 server adapter processing described above. However, each SEND TEXT (ACCUM) vector data area is returned as a separate data area to the service requestor. Any HEADER text would precede these data areas as a separate data area, and any TRAILER data would follow these data areas as a separate data area:

    ```
    (text buffer area)

    01  SNA-TEXT-AREA.
       05  SNA-TEXT-HEADER       PIC X(4096)  VALUE  LOW-VALUES.
       05  SNA-TEXT              PIC X(4096)  VALUE  LOW-VALUES.
       05  SNA-TEXT-TRAILER      PIC X(4096)  VALUE  LOW-VALUES.
    ```
- User transactions that result in outbound Link3270 bridge vectors to exceed 32,000 bytes.

Link3270 server adapters currently only support **1** 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 mechanism.
- Link3270 bridge single transaction mode.

  All interactions with target applications are performed via Link3270 bridge session mode. See *CICS External Interfaces Guide Version 2 Release 2* or higher.

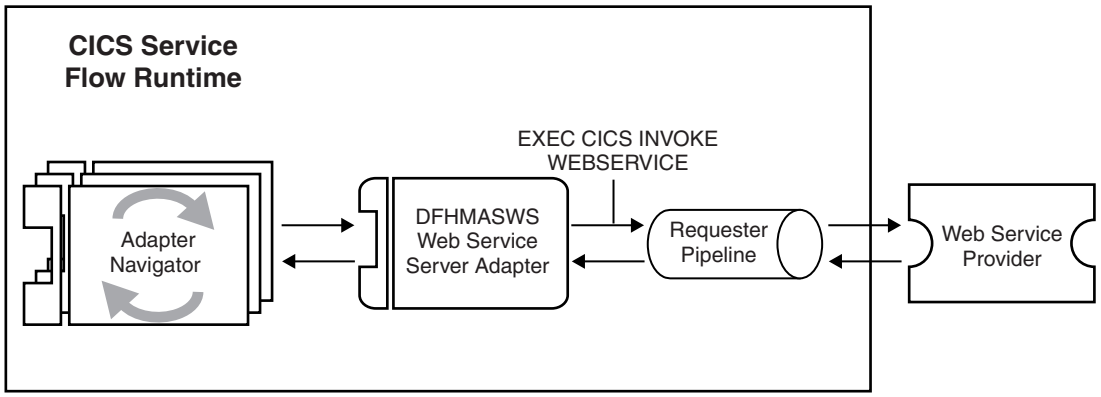
Web services server adapter processing

The Web service server adapter is called DFHMASWS and runs under the CMAO transaction. For it to send outbound web service requests, a suitable PIPELINE and WEBSERVICE resource must be defined in the CICS region.

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 could be on another CICS system or an external provider.

When a response message is received, the pipeline processes this 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 Modeler when the web service request is modeled. The COMMAND.INPUT data-container contains the data to be sent to the Web service provider. The information regarding the target Web Service is stored in the WEBSERVICE.DATA data-container.

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 if 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 whatever action was modeled in the flow.

WebSphere MQ server adapter processing

The processing associated with the WMQ server adapter depends on what is specified for the adapter service and what is passed in the message header from the service requester.

A WMQ PUT server adapter is always generated. The different types of PUT server adapter processing can be specified by the settings of the MsgType field in the WebSphere MQ MQMD structure of the request message header. Depending on what value is specified, a WMQ GET server adapter might not be generated. See “WebSphere MQ node (PUT) properties” on page 277 for details of the settings for the MsgType field.

If you specify a value of 2 for the MsgType field, the Navigator does not:

- Initiate an **MQGET** command to follow the **MQPUT** command.
- Wait on a defined reply queue for a response to the **MQPUT** command.

The WebSphere MQ-enabled application that is the target for the server adapter could be the service requester that issued the original request, or simply an application that performs some work as required as part of the server adapter processing. Whatever role the WebSphere MQ-enabled application plays, it does not need to issue any response to the server adapter. If the target application is the service requester, the **MQPUT** command from the server adapter would be considered an early reply to the request.

The CICS Service Flow Runtime always issues a reply, even when an early reply has been sent from the PUT server adapter. This system reply message could contain only header information, but the application issuing the original request must account for this reply message and retrieve it from the queue. The Navigation Manager only issues replies if you are processing in asynchronous mode and the necessary fields are loaded in the DFHMAH structure.

Processing for non-unique user ids

A non-unique user id is one that is not associated to a single owner, but instead can be used by more than one user. Processing with non-unique user ids assumes that your site is not using them.

The use of non-unique user ids has no impact on Link3270 server adapter business state file (DFHMAL2F) management in CICS Service Flow Runtime V3.1. In addition, the Link3270 node property non-unique user ids is not used and is not required in CICS Service Flow Runtime V3.1. See “Link3270 node properties” on page 280.

For information on LU assignment processing for non-unique user ids using a FEPI adapter, see “FEPI adapter LU assignment processing for non-unique UserIDs” on page 158.

For information on Link3270 bridge facility assignment processing for non-unique user ids, see “Link3270 bridge facility assignment processing for non-unique user ids” on page 366.

Why you should use unique user ids with CICS Service Flow Runtime

Be aware that unique user ids are preferred over non-unique user ids in CICS Service Flow Runtime processing for the following reasons:

- Authentication can be based on user identity — see “Using unique user ids for authentication”
- Secure access to systems — see “Using unique user ids for secure access to systems”
- Potential FEPI performance gains — see “Using unique user ids for potential FEPI performance gains”
- Facilitates problem diagnosis — see “Using unique user ids to facilitate problem diagnosis”

Using unique user ids for authentication

If authentication is required at your site, you can configure the WebSphere MQ-CICS bridge to use the unique user id and an optionally provided 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 the *WebSphere MQ Application Programming Reference* for specific information on MQMD-USERIDENTIFIER.

This userid is used in security authentication in the WebSphere MQ-CICS bridge if configured appropriately. Different levels of authentication can be implemented using a user id and an optionally provided password. See “Considering security” on page 56 for more information.

Using unique user ids for secure access to systems

You can use a unique user id in CICS Service Flow Runtime FEPI processing to signon to back-end systems where security is implemented. You can use the unique user ids in conjunction with a requested FEPI PassTicket to signon to back-end CICS regions where RACF security or another external security manager is implemented.

Using unique user ids for potential FEPI performance gains

Implementing unique user ids allows for additional capabilities with regards to FEPI connections to back-end applications. With unique user ids,CICS Service Flow Runtime FEPI Nodes or SLUs can be used in one request, left assigned to the unique user id, and reused in subsequent request processing. The application may be left in-transaction at a particular screen as modeled. This may achieve performance gains because CICS Service Flow Runtime FEPI sign on processing to back-end systems is not required on each and every request that requires CICS Service Flow Runtime FEPI processing.

Using unique user ids to facilitate problem diagnosis

CICS Service Flow Runtime system errors can be traced to a particular user of the CICS Service Flow Runtime . This type of tracing cannot be done when using non-unique user ids. Also, CICS Service Flow Runtime FEPI connections can be located for a particular user id; facilitating FEPI problem diagnosis and leading to quicker problem resolution.

In addition, CICS Service Flow Runtime Link3270 Vector log file records are written and can be dumped for a particular user id, facilitating Link3270 server adapter problem diagnosis and leading to quicker problem resolution.

Using unique user ids to enhance Link3270 server adapter processing

Implementing unique user ids allows for additional capabilities with regards to Link3270 server adapter processing to back-end applications within MQSI Agent for CICS version 1.1.3. With unique user ids, CICS Service Flow Runtime Link3270 bridge facilities (and 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 may be left in-transaction at a particular screen as modeled.

Note: CICS Service Flow Runtime V3.1 Link3270 server adapter processing provides the benefit described above but does not require the use of unique user ids to achieve that benefit. CICS Service Flow Runtime V3.1 Link3270 server adapter business state data management is performed differently than in MQSI Agent for CICS version 1.1.3. See “Managing state information.”

Managing state information

The CICS Service Flow Runtime manages as much state information as required to support the work to satisfy one invocation. To be clear, understand that this *state information* refers to the business request state data and information.

Once the invocation is satisfied, state information is not retained and is not maintained across multiple invocations of CICS Service Flow Runtime. In the case of a failure, CICS Service Flow Runtime retains state information and application data for subsequent use in a compensation flow. See “How compensation processing works” on page 193 for more information.

The way in which CICS Service Flow Runtime manages state information for Link3270 bridge server adapter request processing varies according to the processing pattern (Aggregate connector persistent or Single connector nonpersistent). The state management processing for each Adapter service type are described in the following sections.

Business state data management in aggregate connectors and single connector (persistent patterns)

For Adapter services of the aggregate connector pattern, both persistent and nonpersistent, and for Adapter services of the single connector persistent pattern, CICS Service Flow Runtime Link3270 server adapter business state data is stored, retrieved and deleted in a CICS VSAM file (DFHMAL2F).

See “Deployment patterns” on page 53 for further information regarding *aggregate connector patterns* and *single connector - persistent* patterns.

The state file DFHMAL2F is mapped by copybook DFHMAR2S in the library*hqual*.SCIZMAC. There are at most two VSAM records for each allocated Link3270 bridge facility. There may be no VSAM records written to store facility business state data if the Link3270 server adapter processing does not require it.

See “Link3270 node properties” on page 280 for information on the deallocate facility indicator, MP-BR-DEALLOCATE-IND, and other Link3270 server adapter properties.

A VSAM file is used to store, retrieve and delete Link3270 facility business state information when the deployment pattern is of the *aggregate* or complex type or when recoverability is a requirement. There can be at most two VSAM records written for each allocated bridge facility 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 would contain a value of 'TEXT".

The VSAM file is used to manage the business state data of facilities and ownership of facilities left allocated. Normally, this file is used to store the last bridge vector received (max. 32000 bytes ADS, text or 3270 data-stream data) from the defined target CICS application transaction as modeled in the sequence 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 and so forth.

Based upon the setting of the *deallocate facility indicator*, in the Link3270 server adapter and the execution completion status of that adapter, the facility used may remain allocated and its associated business state retained upon execution completion. If so, the CICS Service Flow Runtime Link3270 Server Adapter Business State file record is written containing the following:

- allocated facility business state information for that facility owner (LS-USERID = Userid)
- allocated facility token (LS-FACILITYTOKEN = allocated Link3270 bridge facility token)
- (optionally) the service name (LS-SERVICE-NAME = user-defined service name), MP-BR-SERVICENAME) defined in the Properties file.

The facility and it's associated business state is available for subsequent use by the owner of that facility in another task.

The Link3270 bridge facility token is also referred to as the Link3270 facility state token.

Note: Userid is the signed-on user id to the local CICS region as determined by an EXEC CICS ASSIGN command, see the *CICS Application Programming Reference* for further information.

If when Link3270 server adapter processing is complete the Link3270 bridge facility is left allocated with it's associated facility business state data stored in the facility business state file (DFHMAL2F), the Link3270 facility state token is returned to the service requestor/client in the outbound reply message in the DFHMAH header structure field, DFHMAH-STATETOKEN, for use in subsequent request processing if desired. See “DFHMAH header structure” on page 80 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 header structure, DFHMAH, and a Link3270 server adapter is invoked, a new Link3270 facility is allocated and used in Link3270 server adapter processing.

As stated, an allocated Link3270 facility may be de-allocated and it's associated facility business state data deleted when Link3270 server adapter processing is complete based upon the setting of the deallocate facility indicator, MP-BR-DEALLOCATE-IND, for that Link3270 server adapter and the execution completion status of that Link3270 server adapter.

In addition, Link3270 facilities and their associated business state data may be de-allocated and deleted, respectively, by the system cleanup tasks. See “Facility state cleanup processing in the CICS Service Flow Runtime” on page 168 for a description of how the runtime environment performs facility business state VSAM file cleanup.

Also, as a normal end of day processing strategy, the service requestor could invoke a modeled flow to locate any allocated facility, deallocate the facility and delete its associated facility business state data. This flow would need to be run for each allocated Link3270 facility state token by each service requestor invoking runtime services.

Business state data management in single connector (nonpersistent patterns)

In an Adapter service of the *single connector - nonpersistent* pattern, the CICS Service Flow Runtime Link3270 server adapter business state data is stored, retrieved and deleted in a multiple item CICS temporary storage queue (TSQ).

See “Deployment patterns” on page 53 for detailed information on runtime processing of Adapter services of the single connector - nonpersistent deployment pattern.

The Link3270 server adapter business state data is mapped by copybook DFHMAL2C in the library *hqual*.SCIZMAC. There is one TSQ with at most two items for each allocated Link3270 bridge facility. There may not be a TSQ written to store facility business state data if Link3270 server adapter processing does not require it.

See “Link3270 node properties” on page 280 for information on the deallocate facility indicator, MP-BR-DEALLOCATE-IND, and other Link3270 server adapter properties.

TSQs are used to store, retrieve and delete CICS Service Flow Runtime Link3270 facility business state information when compensation is not required. These TSQs can contain two items:

- **Item +1** contains the Link3270 facility business state information used in Link3270 server adapter processing.
- **Item +2** 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 is used to store the last bridge vector received (max. 32000 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 and so forth.

Based upon the setting of the *deallocate facility indicator* in the Link3270 server adapter and the execution completion status of that Link3270 server adapter, the facility used may remain allocated and it's associated business state retained upon execution completion. If so, a CICS Service Flow Runtime server adapter business state TSQ is written containing the allocated facility business state information for that facility owner (SNA-USERID = UserID). The facility and it's associated business state is available for use by the owner of that facility in subsequent request processing.

Note: The Link3270 bridge facility token is also referred to as the Link3270 facility state token.

Note: Userid is the signed-on user id to the local CICS region as determined by an EXEC CICS ASSIGN command, see the *CICS Application Programming Reference* for further information.

If when Link3270 server adapter processing is complete the Link3270 bridge facility is left allocated with it's associated facility business state data stored in the facility business state TSQ, the Link3270 facility state token is returned to the service requestor/client in the outbound reply message in the DFHMAH header structure field, DFHMAH-STATETOKEN, for use in subsequent request processing if desired. See “DFHMAH header structure” on page 80 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 header structure, DFHMAH, and a Link3270 server adapter is invoked, a new Link3270 facility is allocated and used in Link3270 server adapter processing.

As stated, an allocated Link3270 facility may be de-allocated and it's associated facility business state data deleted when Link3270 server adapter processing is complete based upon the setting of the deallocate facility indicator, MP-BR-DEALLOCATE-IND, for that Link3270 server adapter and the execution completion status of that Link3270 server adapter.

In addition, Link3270 facilities and their associated business state data may be de-allocated and deleted, respectively, by the system cleanup tasks. See “Facility state cleanup processing in the CICS Service Flow Runtime” on page 168 for a description of how this cleanup is performed.

Also, as a normal end of day processing strategy, the service requestor could invoke a modeled flow to locate any allocated facility, deallocate the facility and delete it's associated facility business state data. This flow would need to be run for each allocated Link3270 facility state token by each service requestor invoking runtime services.

XML request and response processing

It is recommended that you useWebSphere Developer for System 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 Guide* and can be found in the CICS TS 3.1 Information Center on the Internet at http://publib.boulder.ibm.com/infocenter/cicsts31/index.jsp. For information on using the tooling to perform XML processing, see the WebSphere Developer for System z help that is provided with the tool.

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

XML request and response support consists of two main functions:

1. The XML parse function parses an inbound XML request message and map XML items to a fixed format COMMAREA. See "XML message formats" on page 322 for a sample of a request message in XML format.
2. The XML generate function generates an XML response message from a fixed format COMMAREA.

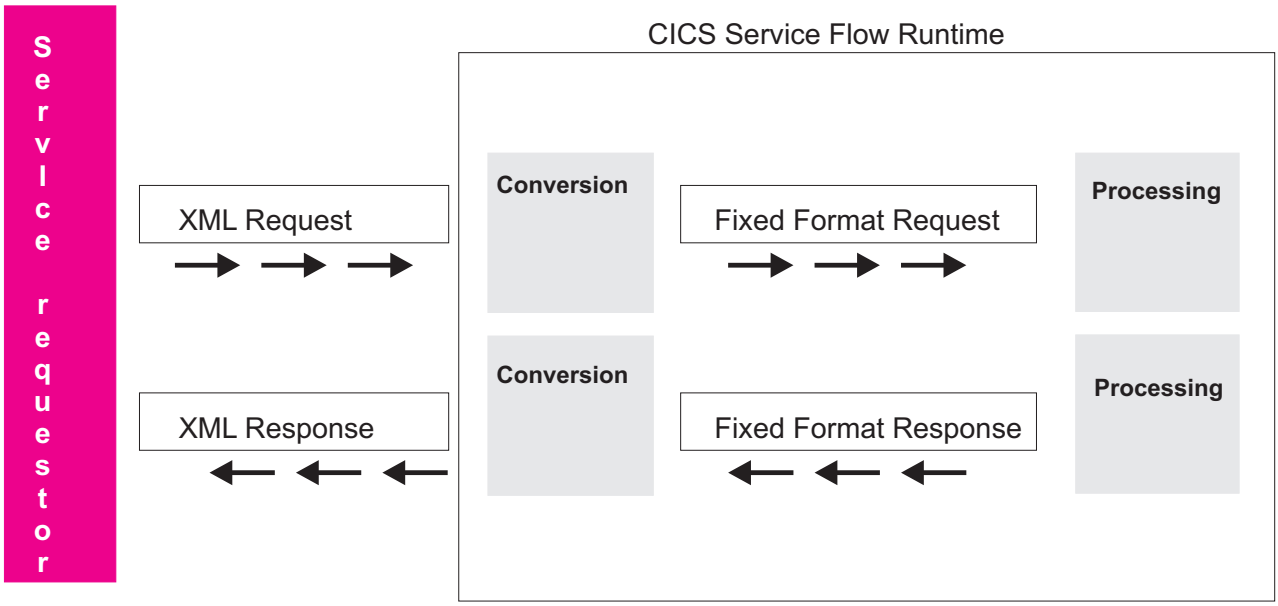


Figure 40. XML request and response processing

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

- The message header (DFHMAH and, if appropriate, DFHMAH2)
- The application data

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

Table 26. Supported XML Message Composition

XML location	Supported Yes or No?
In the message header and in the application data	Yes
In the application data only	Yes

Table 26. Supported XML Message Composition (continued)

XML location	Supported Yes or No?
In the message header only	No

For more information on the structure requirements of the request message, see “Request message headers” on page 80.

See “XML message formats for non-passthrough” on page 322 for samples of XML message formats for non passthrough requests.

XML request processing - non-passthrough

At run time, XML request processing is initiated from the DPL stub program DFHMADPL.

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 Header XML Converter program DFHMAXMI converts the header data from XML to the standard COBOL header format as contained in copybook DFHMAHV.

   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, under the CEEMSG data set, using the message identifier of **IGZ0280S**. The CICS error information is returned to the service requestor.
3. If the application data portion of the request message is also XML, DFHMADPL reads the properties file for the name of the Application Data XML Converter program that converts XML data into the flat COBOL data structure.

   The Application Data XML Converter program is a user defined program. The program name is defined as a property value of the Adapter service, that is specified by the developer at build time using Service Flow Modeler.

   This property value is written to the CICS Service Flow Runtime properties file when the Adapter service is generated and deployed to CICS.

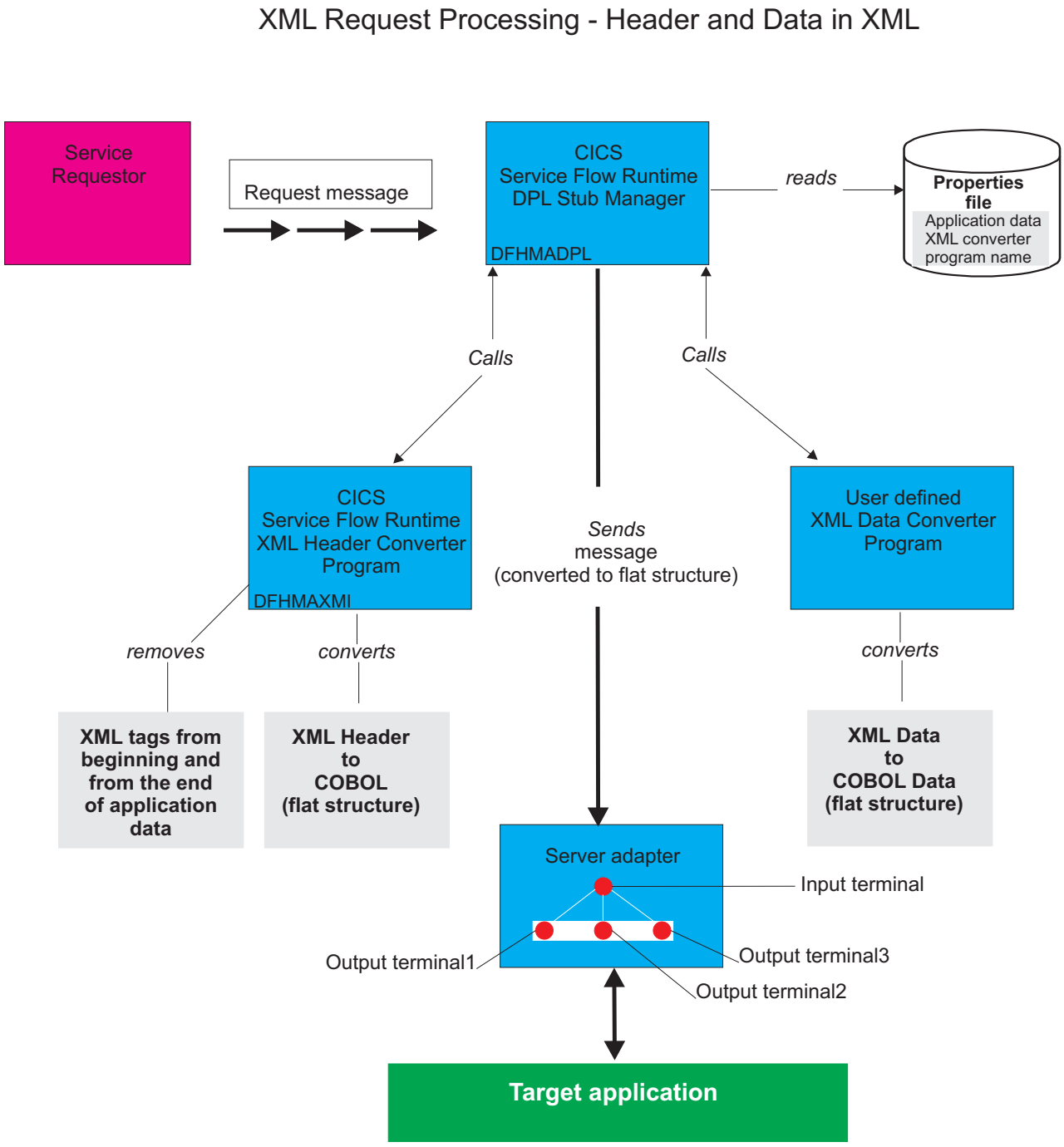


Figure 41. XML Request processing

If the properties for the Adapter service do not include a value for the Application Data XML Converter program, the application data in the request message will not be converted.

If the program call issued by the DPL Stub program to the user defined Application Data XML Converter program fails, CICS Service Flow Runtime writes an error message to be returned to the service requestor.

Standardized call interface to the Header XML Converter program DFHMAXMI

The DPL Stub program invokes the user defined XML data to COBOL conversion program through a standardized call interface that is based on the *XML Enablement of the Enterprise* feature in the WebSphere Developer for System z product.

In addition, the WebSphere Developer for System z product creates a "driver" program that calls the generated converters. The call statement in Figure 42 is adapted from this driver call statement.

```
  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 42. Call to User Defined Converter Program

XML response processing for non-passthrough

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

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

Processing is as follows:

1. The server adapter passes the response data to DFHMADPL.

   Since flow composition allows multiple Output Terminals, the response could be one of several different messages.
2. DFHMADPL constructs a response by performing the following steps:
   a. Calls the Application Data COBOL to XML Converter program.

      The purpose of this program is to place the proper XML tags around the application data that resides in the response message:

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

      The Application Data COBOL to XML Converter program is a user defined program. The program name is specified on the Output Terminal of the flow model, and is generated into corresponding code of the server adapter. This

value is set in a manner consistent will other property values captured in the Service Flow Modeler in the WebSphere Developer for System z product. See the Service Flow Modeler documentation in the WebSphere Developer for System z product for information on setting properties for Adapter services.

If an Application Data COBOL to XML Converter program name is not provided for a reply message, the application data will remain in standard, flat format.

If the program call to the user converter program is unsuccessful, an error message will be returned to the service requestor

The call interface for the Application Data COBOL to XML Converter program is identical to that of the Application Data XML to COBOL converter program. See Figure 42 on page 186.

b. Calls the COBOL to XML Converter program DFHMAXMO. DFHMAXMO performs the following steps:
   1) Converts the structure of the header in the response message from a flat (COBOL) format to an XML format.
   2) Places the XML declaration at the top of the XML message.

      The XML declaration was saved by the XML Header Converter program DFHMAXMI on the inbound request. That declaration was saved by the DPL stub program.
   3) Takes the reply data formatted by the user defined Application Data COBOL to XML Converter program and converts it to XML.
   4) DFHMAXMO reports any errors it encounters to the CICS system log, under the CEEMSG section. Errors written to the CICS system log are returned to the service requestor.

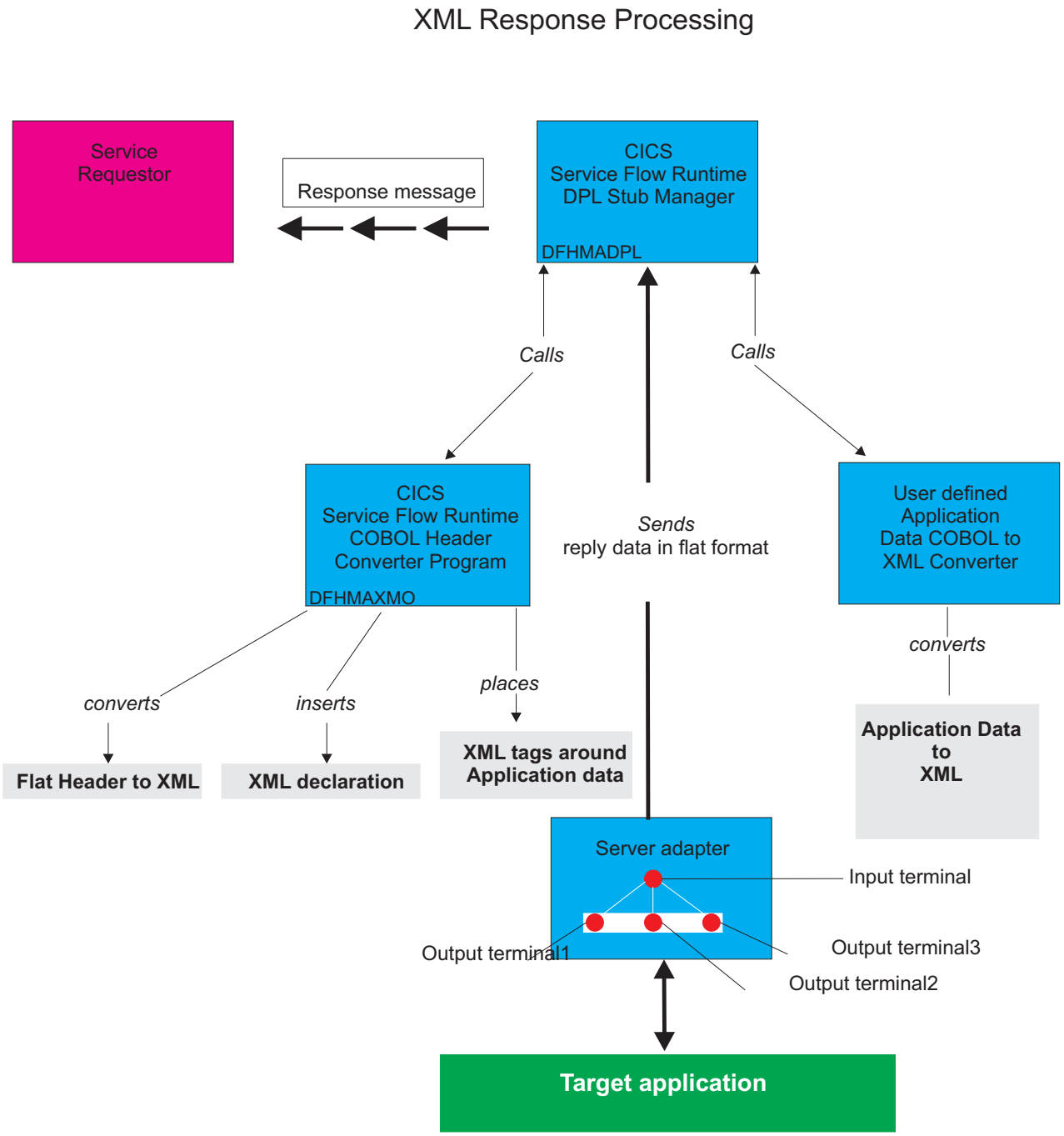


Figure 43. XML Response processing

XML request and response processing for passthrough

The CICS Service Flow Runtime supports the same message composition formats for passthrough requests in XML as it does for non-passthrough requests that are in XML format.

Unlike XML request and response processing for non-passthrough, passthrough requests messages in XML are handled by an Application Data XML Converter program that is provided with CICS Service Flow Runtime.

XML request processing for passthrough

Passthrough request processing is supported by Link3270 server adapters only. At run time, XML request processing for passthrough is initiated from the passthrough stub program (DFHMADPP).

When the passthrough stub program DFHMADPP determines that the request message header is in XML, the following processing occurs:

1. DFHMADPP issues a call to the XML to COBOL Converter program DFHMAXMI.

   DFHMAXMI parses the message header (DFHMAH) and the passthrough message header (DFHMAH2).

2. Instead of using a custom XML parser to parse the application data, the passthrough XML converter program DFHMAXPI is always used to parse the application data in a passthrough request.

   Note: Unlike XML request and response processing for non-passthrough, the Application Data XML Converter program is provided in the runtime environment. XML request and response processing.

   The passthrough XML converter program DFHMAXPI parses the inbound XML passthrough request message (the portion of the XML request message following the header structures, DFHMAH and DFHMAH2), converting the screen header, map header, and screen data from XML to the fixed format screen header, map header, and CICS ADS structure.

   DFHMAXPI references the Link3270 Repository file to create the ADS structure required by the Link3270 bridge mechanism. An XML message format is published to which the service requestor must conform in order to create valid XML request message for Passthrough.

3. DFHMAXPI saves the XML declaration portion of the XML and passes it back to the passthrough DPL Stub program.

   This information is used for XML response processing.

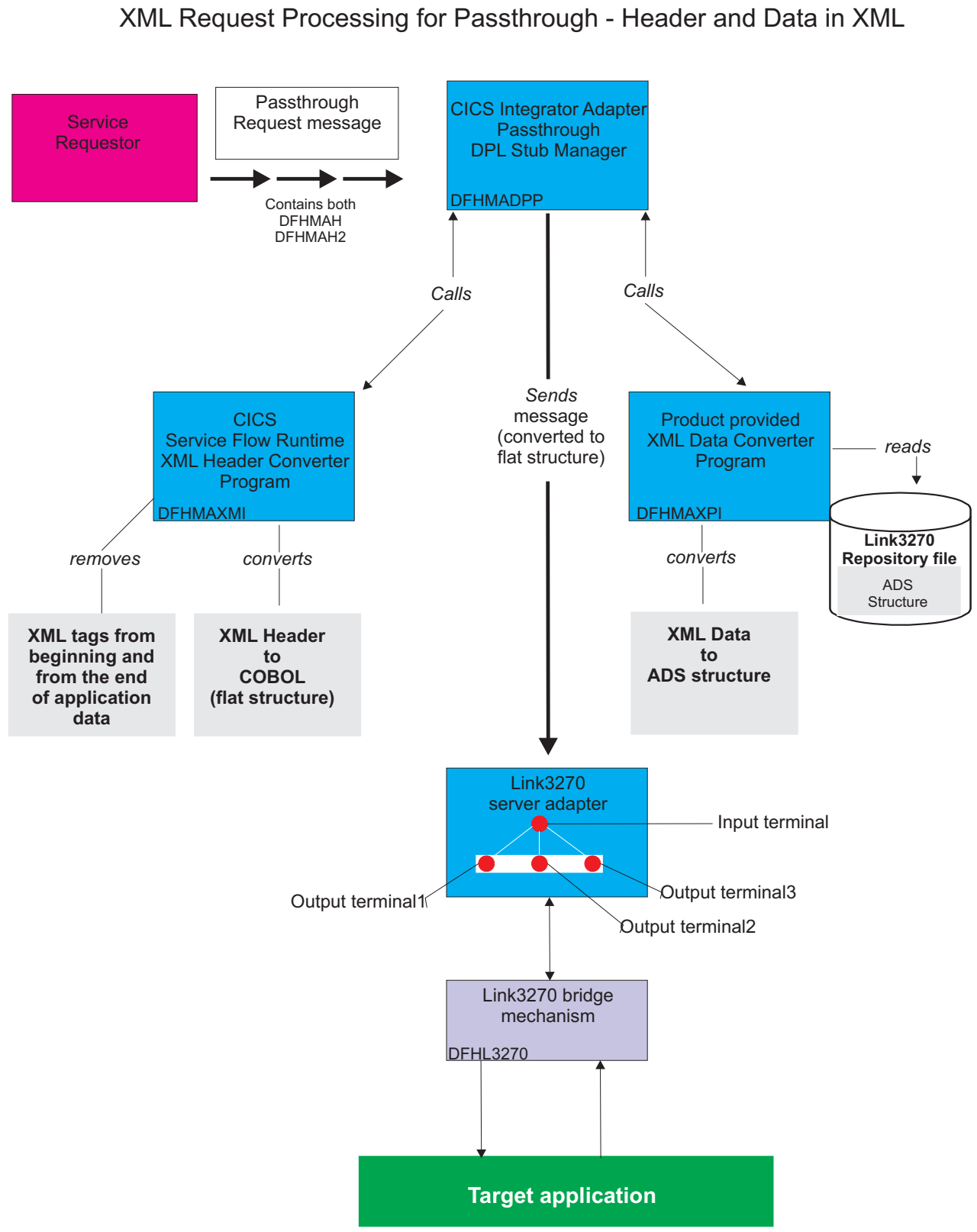


Figure 44. XML Request processing - passthrough

XML response processing for passthrough

At run time, XML response processing is initiated when the Link3270 server adapter passes the reply data from the CICS target application to the CICS Service Flow Runtime DPL Stub program.

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.

Note: The Link3270 server adapter interfaces with the Link3270 bridge mechanism (DFHL3270), which sends vectors to, and receives vectors from, any CICS target application that uses BMS SEND MAP and RECEIVE MAP commands (with some restrictions) where a single SEND MAP and RECEIVE MAP Application Data Structure (ADS).

The CICS Service Flow Runtime passthrough stub program (DFHMADPP) constructs a reply that is consistent with the format of the request message. When the request message is an XML document (i.e. the header in XML format), then the reply is a complete XML document. When only the application data is XML, then the reply is constructed with a standard format header and the application data as an XML document.

Processing is as follows:

1. The Link3270 server adapter passes the response data to the CICS Service Flow Runtime Passthrough Stub program (DFHMADPP).

   Since flow composition allows multiple Output Terminals, the response could be one of several different messages.
2. The CICS Service Flow Runtime Passthrough Stub program (DFHMADPP) constructs a response by performing the following steps:
   a. Calls the CICS Service Flow Runtime COBOL to XML Converter (DFHMAXMO) program. The CICS Service Flow Runtime COBOL to XML Converter (DFHMAXMO) program performs the following steps:
      1) Converts the structure of the header in the response message from a flat (COBOL) format to an XML format.
      2) Places the proper XML tags around the application data that resides in the response message:

         The XML declaration was saved by CICS Service Flow Runtime XML Header Converter program (DFHMAXMI) on the inbound request. That declaration was saved by the CICS Service Flow Runtime DPL stub program.

         ```
         <dfhmaad>
         Application data from response message
         </dfhmaad>
         ```
      3) Takes the reply data formatted by the user defined Application Data COBOL to XML Converter program and converts it to XML.
   b. Calls the CICS Service Flow Runtime Passthrough XML Converter program (DFHMAXPI).

      The call interface CICS Service Flow Runtime Passthrough XML Converter program (DFHMAXPI) is identical to that of the Application Data XML to COBOL converter program. See Figure 42 on page 186.

DFHMAXPI builds the outbound XML passthrough application response message using the application reply data passed to it from the calling system program.

DFHMAXPI performs the following processes:

1) Converts the structure of the header in the response message from flat (COBOL) format to an XML format and converts the ADS structure data to XML.

   Note: The CICS Service Flow Runtime XML Conversion programs (DFHMAXI and DFHMAXO) will report any errors that they encounter to the CICS system log, under the CEEMSG section. Additionally, CICS error information will be returned to the service requestor.

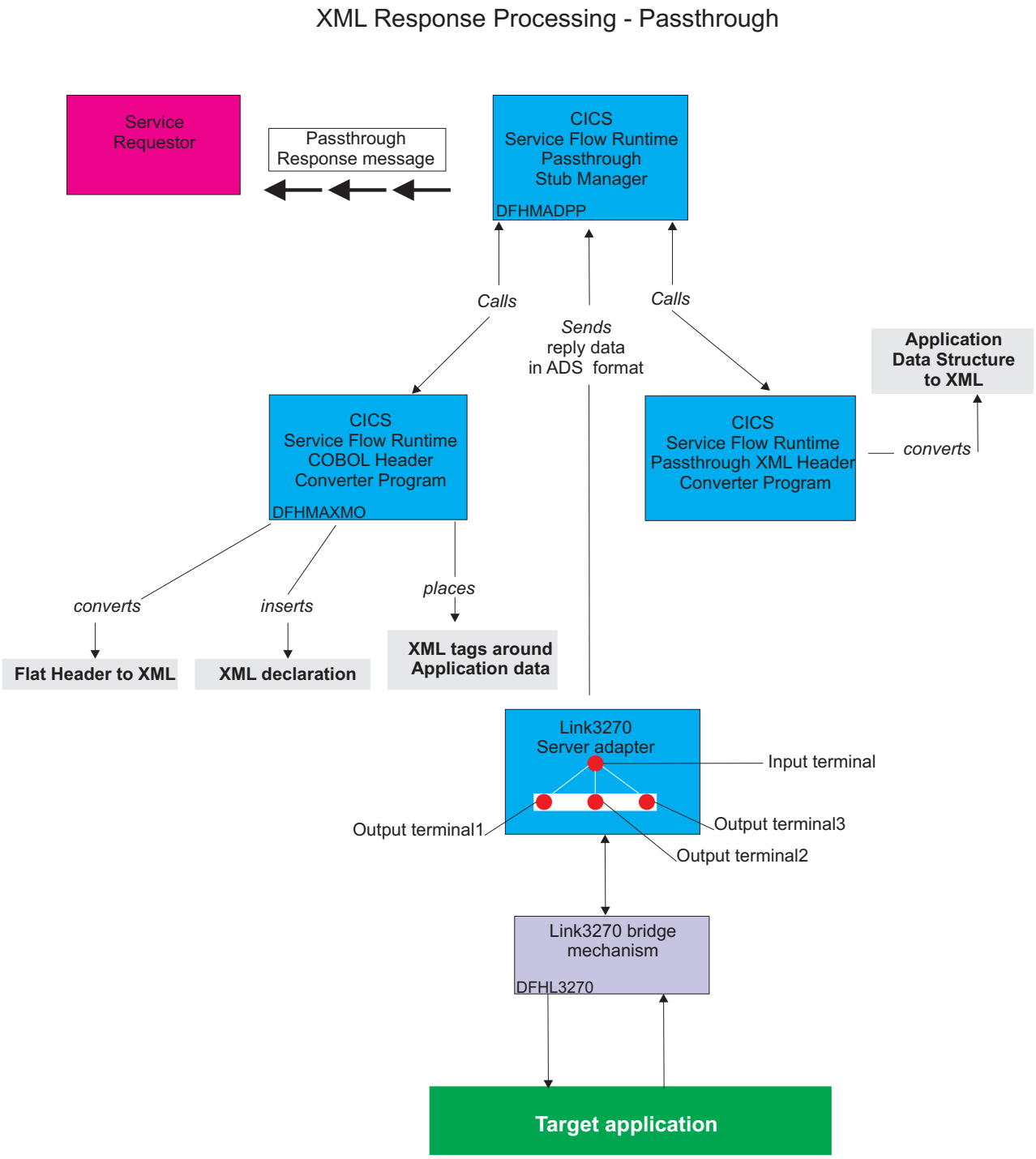


Figure 45. XML Response processing - passthrough

How compensation processing works

At run time, CICS Service Flow Runtime provides compensation control and coordination of adapter request processing. It contains a journaling facility using CICS/BTS container services to support compensation.

Overall responsibility for compensation lies with the service requestor. The run time processing associated with compensation is not initiated automatically when errors occur. Instead, the run time issues a reply (DFHMAH) and it is the responsibility of the service requestor to invoke compensation processing. If the service requestor decides not to invoke compensation processing, the process will remain in CICS SFR until it is cancelled.

Service Flow Modeler does not allow you to explicitly associate a compensating service flow to a modeled service flow. However, a person using the tool can model a service flow for the purpose of being used in compensation, if the service requestor:

- Is programmed with the necessary logic to coordinate the service flow and the compensating service flow.
- Provides the necessary information in the message header (DFHMAH).

When adapter request processing runs in asynchronous mode, syncpoints are taken while the Navigation Manager, Navigator and server adapters complete their processing. These synchronization points provide the necessary state, status and journaling information in the event of subsequent failure. This information can in turn be used in a flow that has been modeled to perform compensation.

The following list provides an overview of how compensation works:

1. The person modeling the sequence flow to be used for compensation needs to set specific parameters and use logic in the service requestor to associate this flow to the one that requires compensation.
2. A service flow that will be used for compensation, must be designated at the highest level (i.e., the Adapter) in the model.
3. If a server adapter fails at run time, state and status information and journal data context are retained. This means that the failed BTS process is left in an incomplete state, therefore allowing other tasks to ACQUIRE the failed process and access the necessary container information. The state and status information is in a process container that is owned by the failed process. The journal context is in a data-container owned by the DFHROOT activity of the failed process (i.e., the Navigation Manager (DFHMAMGR).
4. The failed BTS process name and process type are returned to the service requestor.
5. The service requestor determines whether or not to send a request to invoke compensation. The message structure used by the service requestor adheres to the same structure that a normal request message adheres to. See “Request message headers” on page 80 for a description of structure.
6. The failed BTS process name and process type must be returned in the message structure on compensation. In addition, a new BTS process name and process type must be provided to run the compensating flow. Optionally, compensating application data may be supplied.
7. The DPL Stub program DFHMADPL initiates compensation at run time. DFHMADPL performs the following tasks:
   - ACQUIRE the failed process using the failed process name and process type.
   - Retrieve (issue GET CONTAINER) the state and status and journal information.
   - CANCEL the failed process. At this point the failed process data is unavailable.
   - DEFINE the compensating flow process.

- Write three process containers that contain the state and status information, journal context and compensating input message, respectively. These containers have read-only access to Navigators and server adapters.
- RUN the compensating flow process.

8. As is the case with normal processing, the state and status information is available (in read-only access) to the *compensating* Navigators. The information resides in the ADAPTER.PROCESS process data-container.
9. The journal context is available to *compensating* Navigators in the ADAPTER.JOURNAL process container (again read-only access). It is the modelers responsibility to map from the journal process data-container to the compensating Navigators data context areas or other storage areas if necessary when defining and modeling the service flow to be used in compensation processing. The layout is in DFHMAMAC. The flow must have knowledge of the journal context structure.
10. Any compensating application data is available to *compensating* Navigators as in normal processing in the ADAPTER.INPUT process container, again read-only access.

Error processing

The CICS Service Flow Runtime provides an error log facility that captures *application errors*. Application errors are those errors that the Adapter service detects in the server runtime environment.

Error handling in the CICS Service Flow Runtime is configurable as follows:

- If you define and enable the CMAQ transient data queue (TDQ), the error log facility writes all of the application errors to that defined queue.
- If you have not defined and enabled CMAQ, the error log facility attempts to write the error message CIA08009E to the operator.

Errors that occur outside of the control of the Adapter service are not written to the Error file. For example, the following types of errors would not be written to the Error file:

- Errors in the WebSphere MQ or CICS environments that occur outside of the CICS SFR environment. These errors would be logged within the WebSphere MQ or CICS environments.
- Errors in CICS applications.
- Errors in WebSphere MQ-enabled applications that are run by being invoked by a WebSphere MQ server adapter in the Adapter service.
- Errors in target DPL programs that are run because of being invoked by a DPL server adapter in the Adapter service.

How errors are handled

When the CICS Service Flow Runtime encounters an error, the error records are written to the defined transient data queue (TDQ).

An error listener program retrieves the error records off the queue and writes them to an Error file. The Error Listener program name is DFHMAERQ; the transaction ID is CMAQ. You can also use WebSphere MQ to perform error processing. See “Error processing using WebSphere MQ” on page 372 for details. Both options are shown in the figure below.

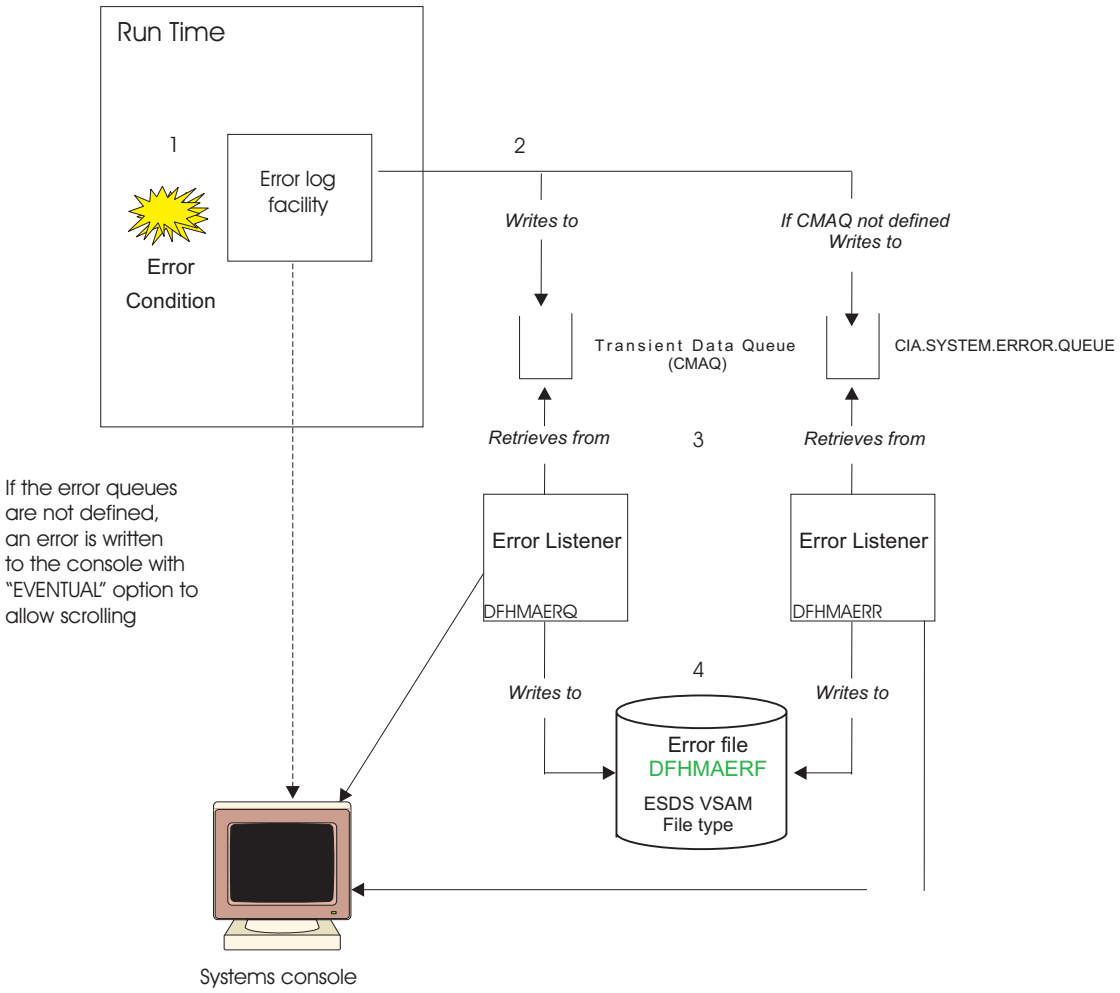


Figure 46. CICS Service Flow Runtime error processing

Error processing using TDQ

Things to know about CICS Service Flow Runtime error processing if you are using the TDQ configuration.

- The queue to which errors are written must be defined and enabled when you customize the CICS Service Flow Runtime.
- If you do not define and enable the error TDQ, CICS SFR writes error message CIA08009E to the console.
- The error listener program DFHMAERQ maps the layout of the error file DFHMAERF using the record description copybook (DFHMARER) found in the **.SCIZMAC** product distribution library. CICS SFR comes with a dump utility for the error file. See "The DFHMAERF error file" on page 211.
- The error TDQ can be predefined with a trigger level and triggered transaction. When the number of entries in the TDQ reaches that level, the triggered transaction is automatically initiated. This results in DFHMAERQ writing the errors out to DFHMAERF. The default trigger level is 1.
- When there is a problem reading or writing messages from or to the error queue, a message that indicates that a problem exists with the particular error queue, is written to the system console. Errors are written to the console with the

EVENTUAL option to allow console scrolling. Errors will need to be deleted from the console by an authorized systems person.

See “Conditions that cause CICS Service Flow Runtime to write to the system console” for more information on when CICS SFR writes a message to the system console and for an example of the format of the messages that are written to the console.

- If you want to perform additional processing of error related information, you can develop your own error listener program to replace the one provided. For example, you could program an *Error Listener* to write error records to other environments (Tivoli®, for example).
- If an error occurs when writing to DFHMAERF, a message is written to the console detailing the problem, and request processing is stopped. The record remains on the CMAQ. When the problem is fixed, the error queue should be enabled and any messages on the queue are drained to DFHMAERF.

Conditions that cause CICS Service Flow Runtime to write to the system console

Messages are written to the system console in the following conditions:

- A problem occurs writing to the transient data error queue CMAQ in the runtime environment during request processing.
- A problem occurs writing a run time processing error (a FILE write error) to the error file DFHMAERF in the Error Listener.

See “Error processing” on page 195 for an explanation of how CICS SFR works to process error messages.

Format of WRITEQ TD error written to console

In this example, the following error:

```
CIA08003E WRITEQ TD error reported in transaction (CMAM) Program (DFHMAMGR) Applid (CICSDEV);
Name (CMAQ) Resp = 84 Resp2 = 0
```

would use this error message format:

```
05  WS-WRITE-TDQ-ERROR.
    10  WS-WRITE-MESSAGE3     PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(02)  VALUE SPACES.
    10  WS-WRITE-TDQ-VERB     PIC X(09)  VALUE SPACES.
    10  FILLER                PIC X(40)  VALUE
        ' error reported in CICS SFR transaction('.
    10  WS-WRITE-TRX3         PIC X(04)  VALUE SPACES.
    10  FILLER                PIC X(11)  VALUE
        ')  Program('.
    10  WS-WRITE-PROGRAM3     PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(10)  VALUE
        ')  Applid('.
    10  WS-WRITE-APPLID3      PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(08)  VALUE
        '); Name('.
    10  WS-WRITE-TDQ-NAME     PIC X(04)  VALUE SPACES.
    10  FILLER                PIC X(10)  VALUE
        '): Resp = '.
    10  WS-WRITE-TDQ-RESP         PIC -9(8)  VALUE ZEROES.
    10  FILLER                PIC X(10)  VALUE
        '  Resp2 = '.
    10  WS-WRITE-TDQ-RESP2    PIC -9(8)  VALUE ZEROES.
```

See the *CICS Application Programming Reference* for an explanation of CICS response codes.

Format of WRITE error written to console

In this example, the following error:

```
CIA01006E WRITE error reported in CICS SFR transaction (CMAE) Program (DFHMAERR) Applid (CICSDEV);
File (DFHMAERF); Resp = 19 Resp2 = 60
```

would use this error message format:

```
05  WS-WRITE-FILE-ERROR.
    10  WS-WRITE-MESSAGE2     PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(02)  VALUE SPACES.
    10  WS-WRITE-FILE-VERB    PIC X(06)  VALUE SPACES.
    10  FILLER                PIC X(40)  VALUE
        ' error reported in CICS SFR transaction('.
    10  WS-WRITE-TRX2         PIC X(04)  VALUE SPACES.
    10  FILLER                PIC X(11)  VALUE
        ')  Program('.
    10  WS-WRITE-PROGRAM2     PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(10)  VALUE
        ')  Applid('.
    10  WS-WRITE-APPLID2      PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(08)  VALUE
        '); File('.
    10  WS-WRITE-FILENAME     PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(10)  VALUE
        '): Resp = '.
    10  WS-WRITE-RESP         PIC -9(8)  VALUE ZEROES.
    10  FILLER                PIC X(10)  VALUE
      '  Resp2 = '.
    10  WS-WRITE-RESP2        PIC -9(8)  VALUE ZEROES.
```

See the *CICS Application Programming Reference* for an explanation of CICS response codes.

Chapter 10. Troubleshooting and support

This section provides information to help you diagnose problems with the runtime environment. It includes 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 an adapter 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 following documentation to assist you with problem determination:

- *CICS Application Programming Guide* and *CICS Problem Determination Guide* for information about using CICS debugging tools, trace, and dump.
- *CICS Business Transaction Services* for detailed information about CICS BTS messages, trace and dump.
- *CICS External Interfaces Guide Version 2 Release 2* and *Version 2 Release 3* or higher for detailed information about the Link3270 bridge mechanism.
- *CICS Front End Programming Interface User's Guide* for detailed information about CICS Front End Programming Interface (FEPI) messages, trace and dump.
- *CICS Messages and Codes* for information on abend codes and CICS system messages.

These recommended CICS manuals are associated with the *CICS Transaction Server for z/OS Release 3.1* library unless otherwise noted . Make sure that you reconcile the level of any manual or guide with the level of the system that your site is running. This same practice should be used for other products that you are using with your application. Problem determination efforts can be hindered by using either obsolete information or information about a level of the product that is not yet installed.

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 conceptual 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 goal is to determine why something does not work as expected and how to resolve the 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 200
- “Where does the problem occur?” on page 200
- “When does the problem occur?” on page 200
- “Under which conditions does the problem occur?” on page 201
- “Can the problem be reproduced?” on page 201

© Copyright IBM Corp. 2001, 2010

The answers to these questions typically lead to a good description of the problem, and that is the best way to start down the path of 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 might seem like a straightforward question; 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 adapter service?
- 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 adapter service or server adapter?

Remember that, even though one area might report the problem, this does not mean that the problem originates 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 within 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, you need to look only as far as the first suspicious event that you find in a diagnostic log; however, this is not always easy to do 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 of day or night?
- 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 this 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 need to occur for the problem to surface?
- 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 adapter services 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 standalone application?

About fixes and updates

If you encounter a problem, first check the list of APARs to see if 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 CICS support site. Selecting this link displays all of 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 this matches your problem. If a fix is available, this 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 should 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 of the fixes (PTFs) that are available. You can then order the ones that you want from that list. This is called a PSP Bucket.

If you think your problem is related to Service Flow Modeler, check the IBM WebSphere Developer for zSeries support site for interim fixes and updates that might solve your problem.

Troubleshooting checklist

The following checklist can help you to identify the source of the problem that is occurring in the runtime environment. By working through the checklist, you should be able to isolate the source of the error.

Recommended responses to specific errors are documented in the applicable sections of “Error messages” on page 220.

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 201 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 258.
   b. If you are following the correct process and still have difficulties, contact IBM Software Support.
3. Are you having problems when running the post-installation jobs or installation verification procedures? For example, you receive a return code other than 0 when a job completes.
   a. For information on diagnosing errors with the post-installation jobs, read “Troubleshooting post-installation errors” on page 203
   b. For information on diagnosing errors with the IVP, read “Responding to IVP errors” on page 36.
4. Does the problem occur when you are updating the properties file? For example, you are running the update in safe mode and there are naming conflicts.
   a. Check the job output for any error messages. This gives you an indication of what problem occurred.
   b. Dump the properties file DFHMAMPF, as described in “Dumping the properties file” on page 99.
   c. Review the definitions in the properties file, as described in “Analyzing the properties file” on page 204, to assess whether you can overwrite the existing definitions.
5. Does the problem occur when you are invoking a deployed adapter service? When a problem occurs in the runtime environment, errors are written to the error file DFHMAERF. You can dump the error file to find out if a particular server adapter is causing the problem.

a. Check that the sample module DFHMAEUP is located in the *hlq*.SCIZSAMP library and has been compiled.
b. Dump the error file using the provided sample JCL job DFHMAMED. The job is also located in the *hlq*.SCIZSAMP library.
c. Analyze the output to determine if a server adapter is causing the problem. The `Error` field contains the message ID. Read the "Error messages" on page 220 to find the type of error that has occurred. Also take note of the `Program` field, as this indicates the program that the error occurred in. This could be the name of a server adapter.

- Is the problem related to a Link3270 adapter? See "Troubleshooting Link3270 server adapters" on page 206
- Is the problem related to the Web services server adapter? See "Troubleshooting the Web services server adapter" on page 208.

6. Optional: Dump and analyze CICS BTS audit trail if it has been configured for use. See "Using a BTS audit trail for problem determination" on page 209 for more information.
7. Debug your applications. See "Debugging your application to facilitate problem determination" on page 209 for more information.
8. Dump and analyze any CICS trace and dump information available. See "Using CICS trace for problem determination" on page 210 and "Using CICS dump for problem determination" on page 210 for more information.
9. Use additional CICS supplied transactions. See "Using CBAM for problem determination" on page 211 for more information.

If the checklist does not guide you to a resolution, you should contact IBM to report a problem. Before you do this, you might need 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 what information to collect and how to send this to IBM, read the Mustgather: Documentation for an SFR problem in CICS on z/OS that is published on the IBM support Web site at http://www-1.ibm.com/support/docview.wss?rs=1083&context=SSGMGV&uid=swg21239511.

Troubleshooting post-installation errors

Read the following information if you are experiencing problems when running the post-installation jobs.

- If you are running the DFHMAINJ sample job with validation enabled and it fails with a return code other than 0:
  1. Check the job output for any messages that are prefixed with CIAI. This will indicate which parameter values have caused a problem.
  2. 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:
  1. Check the job output to see what error messages were issued. This should give you an indication of what caused the job to fail. Update the values in DFHMAINJ.
  2. 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 -
//* datasets before recreating them. -
//* -
//* It is vitally important that an empty SCIZSAMP -
//* dataset 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'
```

3. Enable validation for the job by changing `novalidate` to `validate` on the DFHMAINR invocation statement in the //REXX step.
4. Rerun DFHMAINJ. If you still get a return code other than 0, check the job output for CIAI messages. This will tell you why the job is invalid. Fix the job as appropriate.
5. Comment the delete statements out 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 don't need to include the deletion statements.

- If you are running the DFHMASET job, a return code other than 0 could be returned for some of the steps.
  - If you get a return code of 4, accompanied by error message `IGYDS0001` when compiling DFHMADCD, DFHMADCI, DFHMADUP, DFHMAEUP and DFHMAVUP, this 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 get a return code that is not 0 or 4, you should contact IBM Software Support providing a copy of the DFHMASET job that you used and the CICS job log.

Analyzing the properties file

You can use information in the properties file to help diagnose problems with deployed adapter services.

To view the properties file you need to dump it, as described in “Dumping the properties file” on page 99.

The properties file is called DFHMAMPF and contains definitions for every adapter service that is deployed in CICS Service Flow Runtime. It is mapped by the copybook DFHMARPF.

1. To find the definitions for a particular adapter service, look for the Property type field where the value is `R` and where the Name field value is the request name of the adapter service. Each adapter service has one type R record, which describes various properties of the adapter service and what processing should occur when it is invoked. In the following example, when a service requester passes the request name DEMO_NAV to the DPL stub program, the request is processed asynchronously and is managed by the adapter navigator DEMOFNAV under the DF01 transaction.

```
02/13/06                                CICS SFR Properties file (DFHMAMPF) Dump                PAGE    1
 ------------------------------------------------------------------------------------------------------------------

  Property type:  R (Request properties)       Name:  DEMO_NAV
  Request type:  0 (Async)       Navigator name:  DEMOFNAV     Navigator transid:  DF01
```

For reference information on the request properties fields for an adapter service, see “Request properties” on page 274.

2. After the type R record, there is a record for every server adapter that comprises the adapter service. The property type field can have one of the following values:

   1 (DPL node)
   The definition of a DPL server adapter. The Name field indicates the program name of the server adapter. The Linked-to program field indicates the target program that the DPL server adapter will link to when it is invoked as part of the request processing. The Remote system name field indicates the SYSID of the CICS region where the target application is installed.

```
Property type:  1 (DPL node)                Name:  DFHMAIP2
Linked-to program:  DFHMABP4     Remote system name:  MAD2     Mirror transaction:     SYNCONRETURN:  N
```

   For reference information on the fields for a DPL server adapter, see “DPL node” on page 277.

   2 (WebSphere MQ node (PUT))
   The definition of a WebSphere MQ PUT server adapter. The Name field indicates the program name of the server adapter. The MsgType field indicates what type of message is going to be put on the request queue. In the example below, it is a request message. This means that a WebSphere MQ GET server adapter is also defined for the adapter service to retrieve the response.

```
Property type:  2 (MQSeries node (PUT))          Name:  DFHMAIP3
MQMD MsgType:   1 (Request)      Request queue:  CIA.IVP.DFHMABP6.REQUEST.QUEUE
MQMD ReplyToQ:  CIA.IVP.DFHMABP6.REPLY.QUEUE     MQMD ReplyToQMgr:
```

   For reference information on the fields for a WebSphere MQ PUT server adapter, see “WebSphere MQ node (PUT) properties” on page 277.

   3 (FEPI node)
   The definition of a FEPI server adapter. The Name field indicates the program name of the server adapter. The Pool name field value is the pool name that is used in the **FEPI ALLOCATE POOL** command to establish the connection and conversation with a target application.

```
Property type:  3 (FEPI node)               Name:  DEMOBRW
Pool name:  CICSDEV2     Target name:              Timeout value:  25  seconds   Exit action:  R (Release)
Non-unique user:  N
```

   For reference information on the fields for a FEPI server adapter, see “FEPI node properties” on page 278.

   4 (WebSphere MQ node (GET))
   The definition of a WebSphere MQ GET server adapter. The Name field indicates the program name of the server adapter. The WaitInterval field defines how long the server adapter waits for a message when it issues an **MQGET** command. The ReplyToQ field indicates which message queue has the reply message on it. This should match the value for the same field in the WebSphere MQ PUT server adapter.

```
Property type:  4 (MQSeries node (GET))        Name:  DFHMAIP4
MQGMO WaitInterval:          30  seconds
MQMD ReplyToQ:  CIA.IVP.DFHMABP6.REPLY.QUEUE   MQMD ReplyToQMgr:
```

   For reference information on the fields for a WebSphere MQ GET server adapter, see “WebSphere MQ node (GET) properties” on page 280.

5 (Link3270 node)

The definition of a Link3270 server adapter. The Name field indicates the program name of the server adapter. The Keeptime field indicates how many seconds the bridge facility should be kept for before being deleted. Setting this value ensures that bridge facilities are not reserved for too long. The AOR routing field indicates whether transactions that run in this server adapter can be routed, either dynamically or as defined for the transaction. In the example, this is not enabled for the server adapter. The Vector logging field indicates whether the Link3270 bridge inbound and outbound vectors are logged for this server adapter

```
Property type:  5 (Link3270 bridge node)     Name:  DFHMAIP6
Service name:  (None)       Facilitylike:           Keeptime:   3600  secs     Getwaitinterval:     1  msecs
Deallocate on exit:  1  (Always)             Non-unique userid:  N     AOR routing:  N     Vector logging:  ON
```

For reference information on the fields for a Link3270 server adapter, see “Link3270 node properties” on page 280.

You can also have an UNKNOWN value, which could indicate that the data in that field is not allowed.

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.

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 CIA07001E to CIA07999E, or unexpected or incorrect data is being returned when you invoke the Adapter service, you need to 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.

1. Check the CICS job log for any DFH prefixed error messages. A DFH prefixed message could indicate that there is an underlying problem with the configuration of the CICS region. For example, if you get 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. Dump the error file DFHMAERF using the provided sample JCL.
   a. Check that the sample module DFHMAEUP is located in the *hlq*.SCIZSAMP library and has been compiled.
   b. Use the sample JCL job DFHMAMED to dump the error file. The job is also located in the *hlq*.SCIZSAMP library.
3. Analyze the dump of the error file to see if this provides you with some indication of what the problem is. See “Analyzing the error file dump” on page 207 for what information to look for in the dump.
4. Enable vector logging for your server adapter. You can either:
   - Set vector logging as an option in the flow usingService Flow Modeler, regenerate the Adapter service and redeploy it.
   - Update the record in the DFHMAMPF properties file for the Link3270 server adapter. To do this, edit the TYPE=5 record in the update job DFHMAMPU, changing the value of **PARM08** to 1. Run the update job.
5. Invoke the Adapter service to capture the flow of data through the Link3270 server adapter. This updates the vector log file.

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, which dumps the vector log file. See “Link3270 Vector Log file dump JCL (DFHMAMVD)” on page 271 to view the sample JCL.
7. Analyze the vector log file dump. See “Analyzing the vector log file dump” for what information to look for in the dump.

Analyzing the error file dump

The error file is called DFHMAERF and contains information about any errors that have occurred.

There are two sections in the error file. The first contains some standard information, so no matter what type of error has occurred, the fields in this section will always be present. The second section contains specific information about the nature of the error.

1. Review the first section of the formatted error file dump. This will give you the error message that was issued, the request name that was being processed when the error occurred, the program that was running and other useful information.
   a. Look up the error message that is listed in the Error field. This will indicate what error occurred and the actions the system took. In many cases, there will be some user actions that are described in the message.
   b. Check the Type field to indicate what server adapter the error occurred in. The meanings of the values are listed in “Type: (ERR-REC-PROGRAM-TYPE)” on page 287.

   For a complete list of the fields that are provided in this section of the error file dump, see “Static portion of the formatted error file dump” on page 285.
2. Review the second section of the formatted error file dump. The Error detail field marks the beginning of this section. The fields vary depending on the type of error that occurred, as indicated by the Type field. The fields for each type of error are described in “Dynamic portion of the formatted error file dump” on page 290.
   a. In particular look at the CICS RESP and RESP2 fields. Look up the codes in the *CICS Appplication Programming Reference*. This manual contains the definitions for all of the RESP and RESP2 codes that are returned by CICS commands and should give you a good idea of why there was a failure.
   b. If the error occurred in a Link3270 server adapter, this second section displays the inbound and outbound Link3270 bridge message field values.

Analyzing the vector log file dump

The vector log file dump displays the flow of data between CICS applications and a virtual terminal.

The vector file dump can contain the header structures, the inbound and outbound vectors, and optionally the vector data.

1. Review the header of the vector file dump to find the context of where the error occurred. For example, it will tell 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 is as the result of a prior 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 the codes up in the *CICS External Interfaces Guide Version 2 Release 2*. This will give you an indication of what error occurred.

An annotated example of the vector log file is provided in “Vector file dump” on page 312.

Troubleshooting the Web services server adapter

If you are experiencing a problem with the Web services server adapter, read through the following information to find out how to diagnose the problem.

The Web services server adapter DFHMASWS enables an adapter service to send a Web service request to a service provider using the existing Web services support in CICS. If the error message CIA08112E is in the error file dump or a SOAP fault message, this indicates that an error occurred in the runtime environment when it tried to issue the Web service request.

1. Check the CICS job log for any DFHPI prefixed error messages. A DFHPI prefixed message could indicate that there is an underlying problem with the Web services support in the CICS region. If there are any DFHPI messages, follow the guidance in the message details to fix the problem. You can also check the Diagnosing problems section of the *CICS Web Services Guide*.
2. Analyze the dump of the error file to see if this indicates what the problem is.
   a. Look for the error message CIA08112E. 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:  CIA08112E    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 *CICS Web Services Guide*. In the example above, CICS returned a RESP code of 16 and a RESP2 code of 4, which indicates that the URI is invalid.

3. Fix the problem and invoke the adapter service again to ensure it works correctly. In the example above, you would need to change the URI that is passed to the Web service server adapter. Depending on the nature of the problem, you might need to redeploy your adapter service into CICS. For details on how to do this, see “Updating an existing adapter service” on page 64.

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 an adapter service and run in your CICS systems. This can assist in problem determination.

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.

1. Use the AUDITLOG and AUDITLEVEL attributes of the PROCESSTYPE resource for the adapter service to control the audit logging that takes place and where the logs are stored. Although you can create audit trails in production, you should use caution as this 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

      This 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 perform write audit records to the audit log.
2. To read the records in the audit log, use the sample job DFHMABAP that is located in the SCIZSAMP samples library. This job runs the DFHATUP utility. See “Audit file dump JCL (DFHMABAP)” on page 269 to view the sample JCL.

For detailed information about audit levels and configuration, see the *CICS Business Transaction Services* manual.

Debugging your application to facilitate problem determination

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.

Use EDF to debug your adapters - adapters are CICS applications that use the CICS BTS API.

The sample compile PROC has the `NOEDF` option specified.

As the adapters do not run attached to a terminal, the CEDX transaction must be used specifying the transaction id of the navigator or server adapter that you want to debug.

See the *CICS Application Programming Guide*. In addition, it can be useful to use a 3rd party debug product that can provide a COBOL verb-at-a-time capability.

Using CICS dump for problem determination

Dumps 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 in conjunction 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 the *CICS Problem Determination Guide*.

Using CICS trace for problem determination

CICS provides a tracing facility that enables you to trace transactions through the CICS components as well as 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.

If you want to define tracing when CICS is running, use the CETR transaction. This transaction is described in the *CICS Supplied Transactions* manual.

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. For detailed information about using component codes to set the level of tracing, see the *CICS Problem Determination Guide*. The component codes are:

Domain name	CICS component code
Business application manager	BA
Event manager	EM
Scheduler services	SH

   This will provide 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 will provide you with information about the programs that are running and the contents of the BTS data-containers.
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. This will trace what is happening in the pipeline when a service requester invokes an adapter service from a Web service, or if an adapter service makes an outbound Web service request.

Using CBAM for problem determination

You might also find it useful and even necessary to inquire on and control CICS BTS resources.

Inquiring on a BTS PROCESSTYPE or task using the CEMT transaction, and using the BTS browser transaction CBAM, can 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 master terminal transaction, CEMT, see the *CICS Supplied Transactions* manual. For information on the CBAM transaction and on the useful CEMT commands in a BTS environment, see the *CICS Business Transaction Services* manual.

Troubleshooting aids

To troubleshoot problems with CICS Service Flow Runtime, you can use the following tools and utilities.

The DFHMAERF error file

Errors that are detected by an Adapter service, and written to the CMAQ transient data queue (TDQ), are captured in an error file to help with troubleshooting problems.

The error file contains relevant information related to the processing that was taking place when the error occurred. It includes, but is not limited to, capturing error information for the following components:

- VSAM files
- CICS BTS APIs and BTS data-containers
- CICS APIs
- FEPI
- WebSphere MQ
- Link3270 bridge mechanism
- XML processing and parsing
- Temporary storage queues (TSQ)
- Run time processing and abends

You can access the contents of the error file by dumping it using the provided sample JCL job DFHMAMED. This job runs the dump utility DFHMAEUP, which formats the file so that you can read it to help with problem determination. The dump of the error file contains two sections:

1. A static section, that contains a standard set of fields to provide key information about the error, no matter which type of error has been captured.
2. A dynamic section, that contains specific information related to the type of error that has been captured. The contents of this section can vary.

Errors that occur outside of the control of the Adapter service are not written to the error file. For example, the following types of errors would not be written to the error file:

- Errors in the WebSphere MQ or CICS environments that occur outside of the CICS SFR environment. These errors would be reported in the WebSphere MQ or CICS environments.

- Errors in CICS applications.
- Errors in WebSphere MQ-enabled applications that are invoked by a WebSphere MQ server adapter in the Adapter service.
- Errors in target DPL programs that are invoked by a DPL server adapter in the Adapter service.

Vector logging

Vector logging is a diagnostic tool that allows you to record the flow of data between CICS applications and a virtual terminal. It is designed to be used in a development environment to assist in the deployment and running of a generated Adapter service that includes Link3270 server adapters, as well as passthrough processing.

Vector logging can be set as an option when a flow is modeled in Service Flow Modeler. When the generated Adapter service is deployed, a record is written in the properties file to specify that vector logging should 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, you can update the record for it in the properties file using the update job DFHMAMPF.

There are two levels of granularity that you can select for vector logging:

Full vector logging
: This level of logging records the header structures, the inbound and outbound vectors including any vector data. The vector data could be an inbound or outbound application data structure (ADS), text, or 3270 datastream.

Vector logging trace
: This level of logging records the header structures, as well as the inbound and outbound vectors. It does not record any vector data.

When the Link3270 server adapter is operational in the runtime environment, it writes to a vector log file called DFHMALVF. 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.

If you switch vector logging on, the server adapter continues writing information until the file is full. When the file is full, vector logging stops, even if the server adapter has not finished request processing. If you apply APAR PK32131, the vector logging functionality has improved so that two files, DFHMALVA and DFHMALVB, are used to 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.

The benefit of using two files is that you can analyze the complete vector logging data for a server adapter. It is very unlikely that a server adapter would write enough data to fill both files, but if this does happen, the most recent data is always available.

Passthrough processing

Vector logging for passthrough processing is performed in the same way as the logging for Link3270 server adapters. The only difference is that the vector logging is handled by the Link3270 passthrough manager program DFHMALPT, as there is no Link3270 server adapter.

Passthrough processing does not use the properties file, so you must specify it in the DFHMAH2-VECTOR-LOGGING field of the DFHMAH2 header using Service Flow Modeler. When this field is included in the header, the Link3270 passthrough manager program writes vector logging information to the active vector log file.

Trace points

When you switch auxiliary tracing on to help with diagnosing 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.

The following trace points are provided and each trace point has a number of trace resource ids associated with it.

Point ID	Module	Trace Resource ID	Lvl	Type	Data
AP 0064	DFHMAIVP	CSFRIV01	AP 1	Entry	CIAMAIVP
		CSFRIV02	AP 1	Exit	CIAMAIVP
		CSFRIV03	AP 1	Entry	Initial setup
		CSFRIV04	AP 1	Exit	Initial setup
		CSFRIV05	AP 1	Entry	Process request
		CSFRIV06	AP 1	Exit	Process request
		CSFRIV07	AP 1	Entry	Process request
		CSFRIV08	AP 1	Exit	Process request
		CSFRIV09	AP 1	Entry	Set IVP status waiting
		CSFRIV10	AP 1	Exit	Set IVP status waiting
		CSFRIV11	AP 1	Entry	Check shared resource
		CSFRIV12	AP 1	Exit	Check shared resource
		CSFRIV13	AP 1	Entry	Check Link3270 bridge adapter
		CSFRIV14	AP 1	Exit	Check Link3270 bridge adapter
		CSFRIV15	AP 1	Entry	Check FEPI adapter
		CSFRIV16	AP 1	Exit	Check FEPI adapter
		CSFRIV17	AP 1	Entry	Check DPL adapter
		CSFRIV18	AP 1	Exit	Check DPL adapter
		CSFRIV19	AP 1	Entry	Check WMQ adapter
		CSFRIV20	AP 1	Exit	Check WMQ adapter
		CSFRIV21	AP 1	Entry	Write message to the log
		CSFRIV22	AP 1	Exit	Write message to the log
		CSFRIV23	AP 1	Entry	Check FEPI resource

Point ID	Module	Trace Resource ID	Lvl	Type	Data
		CSFRIV24	AP 1	Exit	Check FEPI resource
		CSFRIV25	AP 1	Entry	Check Link3270 bridge resource
		CSFRIV26	AP 1	Exit	Check Link3270 bridge resource
		CSFRIV27	Exc	INQUIRE FILE failure	Program data
		CSFRIV28	Exc	INQUIRE FILE failure	Program data
		CSFRIV29	Exc	INQUIRE FILE failure	Program data
		CSFRIV30	Exc	INQUIRE FILE failure	Program data
		CSFRIV31	Exc	INQUIRE FILE failure	Program data
		CSFRIV32	Exc	INQUIRE FILE failure	Program data
		CSFRIV33	Exc	INQUIRE FILE failure	Program data
		CSFRIV34	Exc	INQUIRE FILE failure	Program data
		CSFRIV35	Exc	INQUIRE FILE failure	Program data
		CSFRIV36	Exc	LINK PROGRAM failure	DFHMAH user data
		CSFRIV37	Exc	LINK PROGRAM failure	Application data mismatch
		CSFRIV38	Exc	LINK PROGRAM failure	Program data
		CSFRIV39	Exc	LINK PROGRAM failure	DFHMAH user data
		CSFRIV40	Exc	LINK PROGRAM failure	Application data mismatch
		CSFRIV41	Exc	LINK PROGRAM failure	Program data
		CSFRIV42	Exc	LINK PROGRAM failure	DFHMAH user data
		CSFRIV43	Exc	LINK PROGRAM failure	Application data mismatch
		CSFRIV44	Exc	LINK PROGRAM failure	Program data
		CSFRIV45	Exc	LINK PROGRAM failure	DFHMAH user data
		CSFRIV46	Exc	LINK PROGRAM failure	Application data mismatch
		CSFRIV47	Exc	LINK PROGRAM failure	Program data

Point ID	Module	Trace Resource ID	Lvl	Type	Data
		CSFRIV48	Exc	WRITE to TDQ failure	Program data
		CSFRIV49	Exc	Abend CIAY	Program data
		CSFRIV50	Exc	Abend CIAY	Program data
		CSFRIV51	Exc	Abend CIAY	Program data
		CSFRIV52	Exc	Abend CIAY	Program data
		CSFRIV53	Exc	Abend CIAY	Program data
		CSFRIV54	Exc	Abend CIAY	Program data
		CSFRIV55	Exc	Abend CIAY	Program data
		CSFRIV56	Exc	Abend CIAY	Program data
		CSFRIV57	Exc	Abend CIAY	Program data
		CSFRIV58	Exc	Abend CIAY	Program data
AP 0065	DFHMADPL	MDPL01EN	AP 1	Entry	Program data
		LDPL01CL	AP 1	Call	Link3270 Adapter call
		QDPL01CP	AP 1	Call	WMQ Adapter PUT 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
	DFHMADPP	MDPP01EN	AP 1	Entry	Program data
		XDPP01CL	AP 1	Call	Called program ID
		XDPP02CL	AP 1	Call	Called program ID
		XDPP03CL	AP 1	Call	Called program ID
	DFHMAERQ	MERQ01EN	AP 1	Entry	Error queue listener
	DFHMAERR	MERR01EN	AP 1	Entry	WMQ error listener
		QERR01CG	AP 1	Call	WMQ Adapter GET call
		QERR01CP	AP 1	Call	WMQ Adapter PUT call
	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
	DFHMALPT	MLPT01EN	AP 1	Entry	Link3270 passthrough
	DFHMALSC	MLSC01EN	AP 1	Entry	Link3270 facility state cleanup (TSQ)
	DFHMALTM	MLTM01EN	AP 1	Entry	Link3270 AOR route completion
	DFHMALTS	MLTS01EN	AP 1	Entry	Link3270 facility state management program (TSQ)
	DFHMAMGR	MMGR01EN	AP 1	Entry	Navigation manager

Point ID	Module	Trace Resource ID	Lvl	Type	Data
		LMGR01CL	AP 1	Call	Link3270 Adapter call
		QMGR01CP	AP 1	Call	WMQ Adapter PUT call
		XMGR01CL	AP 1	Call	Called program ID
		XMGR02CL	AP 1	Call	Called program ID
	DFHMASDP	DSDP01CL	AP 1	Call	DPL Adapter call
		DSDP02CL	AP 1	Call	DPL Adapter call
		DSDP03CL	AP 1	Call	DPL Adapter call
		DSDP04CL	AP 1	Call	DPL Adapter call
		DSDP05CL	AP 1	Call	DPL Adapter call
		DSDP06CL	AP 1	Call	DPL Adapter call
		DSDP07CL	AP 1	Call	DPL Adapter call
		DSDP08CL	AP 1	Call	DPL Adapter call
	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	MVCP01EN	AP 1	Entry	Link3270 vector processing
		LVCP01CL	AP 1	Call	Link3270 Adapter call
	DFHMAXPI	MXPI01EN	AP 1	Entry	XML passthrough parse and generation
	DFHMAD03	DD0301CL	AP 1	Call	DPL Adapter call
		DD0302CL	AP 1	Call	DPL Adapter call
		DD0303CL	AP 1	Call	DPL Adapter call
		DD0304CL	AP 1	Call	DPL Adapter call
		DD0305CL	AP 1	Call	DPL Adapter call
		DD0306CL	AP 1	Call	DPL Adapter call
		DD0307CL	AP 1	Call	DPL Adapter call
		DD0308CL	AP 1	Call	DPL Adapter call
	DFHMAF04	FF0401CC	AP 1	Call	FEPI Adapter CONVERSE call
		FF0401CR	AP 1	Call	FEPI Adapter RECEIVE call
		FF0401CS	AP 1	Call	FEPI Adapter SEND call
		LL0401CL	AP 1	Call	Link3270 Adapter call
	DFHMAQ03	QQ0301CP	AP 1	Call	WMQ Adapter PUT call
		QQ0302CP	AP 1	Call	WMQ Adapter PUT call
	DFHMAG05	QG0501CG	AP 1	Call	WMQ Adapter GET call
	DFHMABP6	MBP601EN	AP 1	Entry	WMQ Customer information maintenance
		QBP601CG	AP 1	Call	WMQ Adapter GET call

Point ID	Module	Trace Resource ID	Lvl	Type	Data
		QBP601CP	AP 1	Call	WMQ Adapter PUT call
	DFHMABP7	MBP701EN	AP 1	Entry	WMQ Account information maintenance
		QBP701CG	AP 1	Call	WMQ Adapter GET call
		QBP701CP	AP 1	Call	WMQ Adapter PUT call
	DFHMASP1	MSP101EN	AP 1	Entry	Simulator
		QSP101CG	AP 1	Call	WMQ Adapter GETcall
		QSP101CP	AP 1	Call	WMQ Adapter PUT call
AP 0066	DFHMADPL	LDPL01RT	AP 1	Return	Link3270 Adapter return
		QDPL01RP	AP 1	Return	WMQ Adapter PUT return
		MDPL01EX	AP 1	Exit	Module successful exit
		MDPL02EX	AP 1	Exit	Module error exit
		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
	DFHMADPP	MDPP01EX	AP 1	Exit	Module successful exit
		MDPP02EX	AP 1	Exit	Module error exit
		XDPP01RT	AP 1	Return	Called program ID return
		XDPP02RT	AP 1	Return	Called program ID return
		XDPP03RT	AP 1	Return	Called program ID return
	DFHMAERQ	MERQ01EX	AP 1	Exit	Error queue listener exit
	DFHMAERR	QERR01RG	AP 1	Return	WMQ Adapter GET return
		QERR01RP	AP 1	Return	WMQ Adapter PUT return
		MERR01EX	AP 1	Exit	WMQ error listener exit
	DFHMALFC	MLFC01EX	AP 1	Exit	LINK3270 state cleanup file exit
	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
	DFHMALPT	MLPT01EX	AP 1	Exit	Link3270 passthrough exit
	DFHMALSC	MLSC01EX	AP 1	Exit	Link3270 facility state cleanup (TSQ) exit
	DFHMALTM	MLTM01EX	AP 1	Exit	Link3270 AOR route completion exit

Point ID	Module	Trace Resource ID	Lvl	Type	Data
	DFHMALTS	MLTS01EX	AP 1	Exit	Link3270 facility state management program (TSQ) exit
	DFHMAMGR	LMGR01RT	AP 1	Return	Link3270 Adapter return
		QMGR01RP	AP 1	Return	WMQ Adapter PUT 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
	DFHMASDP	DSDP01RT	AP 1	Return	DPL Adapter return
		DSDP02RT	AP 1	Return	DPL Adapter return
		DSDP03RT	AP 1	Return	DPL Adapter return
		DSDP04RT	AP 1	Return	DPL Adapter return
		DSDP05RT	AP 1	Return	DPL Adapter return
		DSDP06RT	AP 1	Return	DPL Adapter return
		DSDP07RT	AP 1	Return	DPL Adapter return
		DSDP08RT	AP 1	Return	DPL 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 return
		MVCP01EX	AP 1	Exit	Link3270 vector processing exit
		MVCP02EX	AP 1	Exit	Link3270 vector processing error exit
	DFHMAXPI	MXPI01EX	AP 1	Exit	XML passthrough parse and generation exit
		MXPI02EX	AP 1	Exit	XML passthrough parse and generation error exit
	DFHMAD03	DD0301RT	AP 1	Return	DPL Adapter return
		DD0302RT	AP 1	Return	DPL Adapter return
		DD0303RT	AP 1	Return	DPL Adapter return
		DD0304RT	AP 1	Return	DPL Adapter return
		DD0305RT	AP 1	Return	DPL Adapter return
		DD0306RT	AP 1	Return	DPL Adapter return

Point ID	Module	Trace Resource ID	Lvl	Type	Data
		DD0307RT	AP 1	Return	DPL Adapter return
		DD0308RT	AP 1	Return	DPL Adapter return
	DFHMAF04	FF0401RC	AP 1	Return	FEPI Adapter CONVERSE return
		FF0401RR	AP 1	Return	FEPI Adapter RECEIVE return
		FF0401RS	AP 1	Return	FEPI Adapter SEND return
		LL0401RT	AP 1	Return	Link3270 Adapter return
	DFHMAQ03	QQ0301RP	AP 1	Return	WMQ Adapter PUT return
		QQ0302RP	AP 1	Return	WMQ Adapter PUT return
	DFHMAG05	QG0501RG	AP 1	Return	WMQ Adapter GET return
	DFHMABP6	QBP601RG	AP 1	Return	WMQ Adapter GET return
		QBP601RP	AP 1	Return	WMQ Adapter PUT return
		MBP601EX	AP 1	Exit	WMQ Customer information maintenance exit
	DFHMABP7	QBP701RG	AP 1	Return	WMQ Adapter GET return
		QBP701RP	AP 1	Return	WMQ Adapter PUT return
		MBP701EX	AP 1	Exit	WMQ Account information maintenance exit
	DFHMASP1	QSP101RG	AP 1	Return	WMQ Adapter GET return
		QSP101RP	AP 1	Return	WMQ Adapter PUT return
		MSP101EX	AP 1	Exit	Program data
AP 0067	DFHMADPL	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
		CDPL01PT	AP 1	Data	DFHMAC-PASSTHRU 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

Point ID	Module	Trace Resource ID	Lvl	Type	Data
		CDPL02AP	AP 1	Data	DFHMAC-ALLPARMS container contents of outbound response
		CDPL02PT	AP 1	Data	DFHMAC-PASSTHRU container contents of outbound response
		CDPL02UD	AP 1	Data	DFHMAC-USERDATA container contents of outbound response
		CDPL02ER	AP 1	Data	DFHMAC-ERROR container contents

Error messages

System errors are returned from the CICS Service Flow Runtime environment to the service requestor. All error message codes contain an 'E' or 'W' suffix to indicate if the message is an **E**rror or a **W**arning.

Detailed system error information is written to the error file DFHMAERF. The following system errors are generated.

CICS Service Flow Runtime VSAM file error messages

These error messages are prefaced by **CIA010xx**. You can dump the error log to determine the failed program name, transaction, file name, CICS response codes and key values among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *CICS Application Programming Reference* for CICS response codes and definitions.

CIA01001E: FILE-READ-ERRMSG
Explanation

An EXEC CICS READ command to a VSAM file has failed in a CICS Service Flow Runtime system module, or one of your generated adapter navigators, Link 3270 navigators, FEPI navigators or server adapters. Any generated CICS SFR Link3270 navigator file read error is reported by called system modules DFHMALNM or DFHMALFS.

This error is most likely to occur on the CICS SFR Properties file DFHMAMPF.

The error can also occur if an attempt is made to use a prior Link3270 bridge facility and associated Link3270 business state data. TheCICS SFR Link3270 business state file system cleanup modules, DFHMALFC and/or DFHMALFD, may have executed to delete expired Link3270 facilities and associated state. It is not recommended that the Link3270 system cleanup tasks be executing while performing request processing. The Link3270 business state is stored in file DFHMAL2F.

Note: Prior Link3270 business state is stored in file DFHMALSF.

For more information on facility state cleanup processing, see “Facility state cleanup processing in the CICS Service Flow Runtime” on page 168.

User response

Dump the error log DFHMAERF to determine the problem.

If the error occurred on the CICS Service Flow Runtime Properties file DFHMAMPF, dump the Properties file using job DFHMAMPD to determine if a record exists for the key value used. If not, ensure the CICS SFR Properties file update ran successfully and that the file is defined correctly and enabled to CICS.

If the error occurred on the CICS SFR Link3270 business state file DFHMAL2F, ensure that the CICS SFR Link3270 state cleanup tasks are not executing. You can use the CICS transaction ID, field **Tranid:**, found in the error log dump to determine the program name of the generated Link3270 navigator that called the system module reporting the problem if necessary.

Ensure that the Link3270 business state file is defined correctly and enabled to CICS. A not found condition (NOTFND) on the Link3270 business state file will not cause this error message: CICS response code = 13.

CIA01002E: FILE-RECTYPE-ERRMSG
Explanation

A read to the CICS Service Flow Runtime Properties file, DFHMAMPF, was successful; however, there is a record type mismatch. This means that the record type found on the Properties file did not match the expected value.

User response

Determine the failed program name, the key used to perform the read, the record type of the failed program and the properties record type read. The program name can be found in the DFHMAH structure issued in the reply from CICS SFR to the service requestor or in the error log DFHMAERF. The key value can be found in the error log. To determine the record type of the properties record, you must dump the Properties file and locate the record in question and compare the properties record name equal to the key value used.

If the failed program name found is that of a CICS SFR module, i.e., DFHMAxxx, the key used to read the Properties file is the value of the request name found in the DFHMAH header structure, field **DFHMAH-REQUESTNAME**. The properties record type must be an R. If the program name is that of a generated adapter navigator or of a command, the key used to read the CICS SFR Properties file is that program name value. The record type is dependent on the type of program performing the read. Record type values are as follows:

- +1 = DPL command
- +2 = MQ Put command
- +3 = FEPI Navigator
- +4 = MQ Get command
- +5 = Link3270 Navigator

CIA01003E: FILE-STARTBR-ERRMSG
Explanation

An **EXEC CICS STARTBR** command to a VSAM file has failed in a CICS Service Flow Runtime system module or has failed in one of your generated FEPI navigator programs.

If reported by a generated FEPI navigator, a start browse with the GENERIC and EQUAL options specified to the alternate index SLU Connection file, DFHMAC1F, has failed.

If reported by CICS SFR system module DFHMALFC, a start browse with the GTEQ option specified to the Link3270 business state file DFHMAL2F has failed. The Link3270 business state file system cleanup modules, DFHMALFC, may have executed to delete expired Link3270 facilities and associated state while performing request processing. This is not recommended. For more information on facility state cleanup processing, see “Facility state cleanup processing in the CICS Service Flow Runtime” on page 168.

User response

Dump the error log DFHMAERF to determine the problem. Check the file attributes and that the file is defined correctly and enabled to CICS. A not found condition (NOTFND) will not cause this error message: *CICS response code = 13.*

CIA01004E: FILE-READNXT-ERRMSG
Explanation

An **EXEC CICS READNEXT** command to a VSAM file has failed in a CICS Service Flow Runtime system module or has failed in one of your generated FEPI navigator programs.

If reported by a generated FEPI navigator, a browse of the alternate index SLU Connection file DFHMAC1F, has failed.

If reported by CICS SFR system module DFHMALFC, a start browse with the GTEQ option specified to the Link3270 business state file DFHMAL2F has failed. The Link3270 business state file system cleanup module, DFHMALFC, may have executed to delete expired Link3270 facilities and associated state while performing request processing. This is not recommended. For more information on facility state cleanup processing, see “Facility state cleanup processing in the CICS Service Flow Runtime” on page 168.

User response

Dump the error log DFHMAERF to determine the problem. Check the file attributes and that the file is defined correctly and enabled to CICS. A not found (NOTFND) condition or a duplicate key (DUPKEY) condition will not cause these error messages: *CICS response code = 13 and 15*.

CIA01005E: FILE-REWRITE-ERRMSG
Explanation

An **EXEC CICS REWRITE** command to a VSAM file has failed in a CICS Service Flow Runtime system module or in one of your generated FEPI navigator or Link3270 navigator programs.

This condition can occur on the Target Interaction file DFHMATIF, or the SLU Connection file DFHMACOF if reported by a generated FEPI navigator. If reported by the system module DFHMALFS, this condition occurred on the Link3270 business state file DFHMAL2F.

User response

Dump the error log DFHMAERF to determine the problem. Check the file attributes and that the file is defined correctly and enabled to CICS. A duplicate key (DUPKEY) condition on the alternate index for the SLU Connection file will not cause this condition: *CICS response code = 15*.

If the error occurred on the CICS SFR Link3270 business state file DFHMAL2F, you can use the CICS transaction ID, field **Tranid:**, found in the error log dump to determine the program name of the generated Link3270 navigator that called the system module reporting the problem if necessary.

CIA01006E: FILE-WRITE-ERRMSG
Explanation

An **EXEC CICS WRITE** command to a VSAM file has failed in a CICS Service Flow Runtime system module or has failed in one of your generated FEPI navigator or Link3270 navigator programs.

This condition can occur on the Target Interaction file DFHMATIF, or the SLU Connection file DFHMACOF if reported by a generated FEPI navigator. If reported by CICS SFR system module DFHMALFS, this condition occurred on the Link3270 business state file DFHMAL2F.

Note: Prior CICS SFR generated Link3270 navigator file rewrite errors are reported by the generated Link3270 navigator program.

This condition can also occur in the CICS SFR Error Listener system module (DFHMAERR or DFHMAERQ, depending upon your installation) when attempting to write to the error log DFHMAERF. In this case, the error message is written to the system console.

User response

Dump the error log DFHMAERF to determine the problem. Check the file attributes and that the specified file is defined correctly and enabled to CICS. A duplicate key (DUPKEY) condition on the alternate index SLU Connection file (DFHMAC1F), will not cause this condition: *CICS response code = 15*.

If the error occurred on the Link3270 business state file DFHMAL2F, you can use the CICS transaction ID, field **Tranid**:, found in the error log dump to determine the program name of the generated Link3270 navigator that called the system module reporting the problem if necessary.

CIA01007E: FILE-ENDBR-ERRMSG
Explanation

An **EXEC CICS ENDBR** command to a VSAM file has failed in a CICS Service Flow Runtime system module or in one of your generated FEPI navigator programs.

If reported by a generated FEPI navigator, an end browse on the alternate index SLU Connection file DFHMAC1F has failed.

If reported by CICS SFR system module DFHMALFC, an end browse on the Link3270 business state file DFHMAL2F has failed.

User response

Dump the error log DFHMAERF to determine the problem.

If the error occurred on the CICS SFR Link3270 business state file DFHMAL2F, you can use the CICS transaction ID, field **Tranid:**, found in the error log dump to determine the program name of the generated Link3270 navigator that called the system module reporting the problem if necessary.

CIA01008E: FILE-DELETE-ERRMSG
Explanation

An **EXEC CICS DELETE** command to a VSAM file has failed in a CICS Service Flow Runtime system module or one of your generated Link3270 navigator programs.

If reported by CICS SFR system modules DFHMALFS, DFHMALFC or DFHMALFD, this condition occurred on the Link3270 business state file DFHMAL2F.

User response

Dump the error log DFHMAERF to determine the problem. Check the file attributes and that the specified file is defined correctly and enabled to CICS.

If the error occurred on the CICS SFR Link3270 business state file DFHMAL2F, you can use the CICS transaction ID, field **Tranid:**, found in the error log dump to determine the program name of the generated Link3270 navigator that called the system module reporting the problem if necessary.

Temporary storage queue (TSQ) error messages

These error messages are prefaced by **CIA013xx**. You can dump the error log to determine the failed program name, transaction, TSQ name and CICS response codes among other pertinent information. Use this information to assist in problem diagnosis.

Currently CICS Service Flow Runtime uses temporary storage queues (TSQs) for Link3270 server adapter processing only. Additionally, TSQs are only in use when processing adapter services of the single connector nonpersistent type. Any TSQ error messages are reported in modules, DFHMALTS, DFHMALSC and DFHMALFD.

If the error occurred in a Link3270 navigator, you can use the CICS transaction ID, field **Tranid:**, found in the error log dump to determine the program name of the generated Link3270 navigator that called the system module, DFHMALTS, reporting the problem if necessary.

TSQs are used to store, retrieve and delete CICS SFR Link3270 facility business state information when compensation is not required. These TSQs contain two items:

- **Item +1** contains the Link3270 facility business state information used in Link3270 server adapter processing.
- **Item +2** 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 16 byte TSQ QNAME format is:

```
'DFHMA' + 8 byte bridge facility token + 3 byte x 'FFFFFF'
```

See “Facility state cleanup processing in the CICS Service Flow Runtime” on page 168 for a description of how CICS Service Flow Runtime performs facility business state information cleanup.

CIA01331E: 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 in the CICS Service Flow Runtime” on page 168 for description of how facility state information is managed.

User response

Dump the CICS SFR Error log DFHMAERF to determine the problem.

A queue not found (QIDERR) condition will not cause this error message: *CICS response code = 44.*

CIA01332E: 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.

User response

Dump the Error log DFHMAERF to determine the problem.

CIA01333E: 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

Dump the Error log DFHMAERF to determine the problem.

CIA01334E: 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.

User response

Dump the Error log DFHMAERF to determine the problem.

A queue not found (QIDERR) or (LOCKED) condition will not cause this error message: *CICS response code = 44 and 100 respectively*.

CIA01335E: TS-INQSTART-ERRMSG
Explanation

An INQUIRE TSQNAME START AT command has failed in the CICS Service Flow Runtime Link3270 State Cleanup program (DFHMALSC). The START AT value is 'DFHMA' + LOW-VALUES.

User response

Dump the Error log (DFHMAERF) to determine the problem.

CIA01336E : TS-INQNEXT-ERRMSG
Explanation

An **INQUIRE TSQNAME** command with the NEXT option has failed in the CICS Service Flow Runtime Link3270 State Cleanup program DFHMALSC, with a condition other than END.

User response

Dump the Error log DFHMAERF to determine the problem.

CIA01337E: TS-INQEND-ERRMSG
Explanation

An **INQUIRE TSQNAME** command with the END option has failed in the CICS Service Flow Runtime Link3270 State Cleanup program DFHMALSC.

User response

Dump the Error log DFHMAERF to determine the problem.

Data-container error messages

These error messages are prefaced by **CIA020xx**. The most likely cause of CICS Service Flow Runtime data-container errors is that there is a problem with the BTS Repository file. You can dump the error log to determine the failed program name, transaction, container name, container owner and CICS response codes among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *CICS Business Transaction Services* manual for CICS response codes and definitions.

CIA02001E: GET-CONTAINER-ERRMSG
Explanation

An EXEC CICS GET CONTAINER command has failed.

User response

Dump the error log (DFHMAERF) to determine the problem.

CIA02002E: SET-CONTAINER-ERRMSG
Explanation

An EXEC CICS SET CONTAINER command has failed

User response

Dump the error log (DFHMAERF) to determine the problem.

CIA02003E: PUT-CONTAINER-ERRMSG
Explanation

An EXEC CICS PUT CONTAINER command has failed.

User response

Dump the error log (DFHMAERF) to determine the problem.

CIA02004E: DELETE-CONTAINER-ERRMSG
Explanation

An EXEC CICS DELETE CONTAINER command has failed.

User response

Dump the error log (DFHMAERF) to determine the problem.

DPL server adapter error messages

These error messages are prefaced by **CIA030xx**. You can dump the error log to determine the failed program name, transaction, DPL target program name, target SYSID and CICS response codes among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *CICS Application Programming Reference* for details of the **EXEC CICS LINK** command and CICS response codes.

CIA03001E: DPL-ERRMSG
Explanation

An EXEC CICS LINK command has failed. The most likely cause of this error is a PGMIDERR or SYSIDERR.

User response

Dump the error log (DFHMAERF) to determine the problem. Check to see if the DPL target program name and SYSID are correct. They are specified on your type 1 record PARM01 and PARM02, respectively, for the program reporting the error in the Properties file update JCL. You can dump the Properties file, DFHMAMPF, using job DFHMAMPD.

CIA3002E DPP-ERRMSG

Explanation: An **EXEC CICS LINK** command for DPL server adapter program DFHMADPP has failed.

System action: The DFHMADPL server adapter terminated.

User response: The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes. Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

FEPI server adapter error messages

These error messages are prefaced by **CIA040xx**. You can dump the error log to determine the failed program name, transaction, target name, target APPLID, conversation Id, CICS response codes and External Security Manager (ESM) return codes among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *CICS Front End Programming Interface User's Guide* for CICS response codes and definitions. Additional diagnostic information can be found in the FEPI message log or any event-handling transaction that can be installed.

CIA04001E: PASSTICKET-ERRMSG
Explanation

An EXEC CICS FEPI REQUEST PASSTICKET command has failed. The most likely cause of this is an ESM error.

User response

For response and reason codes returned by RACF, see the *OS/390 Security Server (RACF) Messages and Codes* manual.

CIA04002E: 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 attempting to establish a new conversation with a target application, ALLOCATE POOL, the most likely reason this message will occur is the POOL or TARGET or both are unavailable, not known or there is no session available and in service.

User response

Dump the error log (DFHMAERF) to determine the problem. Check to see if the POOL or TARGET or both are defined, installed and in service. Also check to see if there are available in service sessions for the defined TARGET.

CIA04003E: 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 session lost or other session state violation.

User response

Dump the error log (DFHMAERF) to determine the problem. Dump the error log to determine the problem. A CICS RESP2 value of 50 or 220 will not cause this error message.

CIA04004E: RECEIVE-ERRMSG
Explanation

An EXEC CICS FEPI RECEIVE DATASTREAM command has failed. The command options used are UNTILCDEB and MAXFLENGTH(3600). The most likely cause of this error is a session lost or other session state violation.

User response

Dump the error log (DFHMAERF) to determine the problem. Insure the maximum length of the data returned is not greater than 3600 bytes.

CIA04005E: 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 to retrieve the POOL, TARGET, NODE and SENSEDATA. The most likely cause of this error is that a session has been lost.

User response

Dump the error log (DFHMAERF) to determine the problem. Insure the maximum length of the data returned is not greater than 3600 bytes.

CIA04006E: FREE-ERRMSG
Explanation

An EXEC CICS FEPI FREE command has failed. The option used on this command depends on the value of field SNA-RELEASE-LU-IND in your generated FEPI Navigator. This field can have the following values:

- R = RELEASE
- F = FORCE
- P = PASS
- H = HOLD

This value is derived from your type 3 record PARM04 for the program name reporting the error in the CICS Service Flow Runtime Properties file update JCL (DFHMAMUP).

User response

Dump the error log (DFHMAERF) to determine the problem. To determine the value of field SNA-RELEASE-LU-IND, you can dump the CICS Service Flow Runtime Properties file (DFHMAMPF), using job DFHMAMPD.

CIA04007E: SET-ERRMSG
Explanation

An EXEC CICS FEPI SET DATASTREAM command has failed.

User response

Dump the error log (DFHMAERF) to determine the problem.

CIA04008E: 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.

User response

Dump the error log (DFHMAERF) to determine the problem.

CIA04009E: INQUIRE-ERRMSG
Explanation

An **EXEC CICS FEPI INQUIRE POOL** command has failed. The most likely cause of this error is the POOL name is not known. Make sure the POOL is defined and installed.

User response

Dump the error log (DFHMAERF) to determine the problem. You can also check to insure the POOL specified on your type 3 record PARM01 for the program reporting the error in the CICS Service Flow Runtime Properties file update JCL is correct. You can dump the CICS SFR Properties file DFHMAMPF using job DFHMAMPD.

CIA04010E: MAP3270-ERRMSG
Explanation

A severe error has occurred in assembler module DFHMAFS0 entry point BLD3270I when attempting to build an inbound datastream.

User response

Dump the error log (DFHMAERF) to determine the problem.

CIA04011E: PROPERTY-ERRMSG
Explanation

The inquired POOL name has definitions that are not allowed or is not in the correct state for use in the identified FEPI Navigator program. This can be caused by any of the following conditions:

- A FORMAT definition is not equal to DATASTREAM
- A MAXFLENGTH definition is greater than 3600 bytes
- The install state, INSTLSTATUS, is not equal to INSTALLED
- The service state, SERVSTATUS, is not equal to INSERVICE

User response

Check the POOL definitions and state. You can check the POOL definitions by inquiring on the associated installed PROPERTYSET. By using the CEMT supplied transaction, you can view the state of the identified POOL via CEMT INQ FEPOOL command.

See the FEPI INQUIRE POOL command in the *CICS Front End Programming Interface User's Guide (FEPI)* for more information.

WebSphere MQ call error messages

These error messages are prefaced by **CIA050xx**. You can dump the error log to determine the failed program name, transaction, queue name, queue manager name, completion codes and reason codes among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *WebSphere MQ Application Programming Reference* for completion and reason code values and definitions.

CIA05001E: MQOPEN-ERRMSG

Explanation

A call to MQOPEN has failed in either the Error Listener program (DFHMAERR) or in a generated MQ Get command program. Access options used are MQ00-INPUT-AS-Q-DEF and MQOO-INQUIRE.

User response

Dump the error log (DFHMAERF) to determine the problem. If the error is reported by the Error Listener program (DFHMAERR), an attempt was made to open the error queue, CIA.SYSTEM.ERROR.QUEUE. Check to insure the queue manager connection exists and that the queue manager is not quiescing.

You may also want to check the queue definition to ensure that the queue is defined SHARE and DEFSOPT(SHARED) if more than one process may be accessing the queue at the same time.

If the error is reported by a generated MQ Get command program, also insure that the queue the program is attempting to open exists. The queue and queue manager names are specified on your type 4 record PARM02 and PARM03, respectively, for the program reporting the error in the CICS Service Flow Runtime Properties file update JCL. You can dump the Properties file (DFHMAMPF) using job DFHMAMPD.

CIA05002E: MQPUT1-ERRMSG

Explanation

A call to MQPUT1 has failed. This error may be reported on a Put to the ReplyToQ specified in the DFHMAH header by system modules DFHMADPL and DFHMAMGR when attempting to issue reply messages to the service requestor. It also may occur in your generated MQ Put command program and on Puts to the CICS Service Flow Runtime error queue, CIA.SYSTEM.ERROR.QUEUE. The Put options used on Puts to the error queue and Puts issued in your generated MQ Put command program when processing in synchronous mode are MQPMO-NO-SYNCPOINT and MQPMO-DEFAULT-CONTEXT. All other MQPUT1 commands are specified with the Put options equal to MQPMO-SYNCPOINT and MQPMO-DEFAULT-CONTEXT.

User response

Check the queue attributes, such as PUT(DISABLED), CURDEPTH and MAXMSGL first. Dump the error log (DFHMAERF) to determine the problem. If the error is reported by a generated MQ Put command program, insure the queue that the program is attempting to put a message to exists. The queue name is specified on your type 2 record PARM02 for the program reporting the error in the Properties file update JCL. You can dump the Properties file, DFHMAMPF, using job DFHMAMPD.

CIA05004E: MQGET-ERRMSG
Explanation

A call to MQGET has failed or returned a warning in either the CICS Service Flow Runtime Error Listener system module (DFHMAERR) or a generated MQ Get command program. The following Get options are used on the MQGET command when attempting to read messages off the error queue CIA.SYSTEM.ERROR.QUEUE:

- MQGMO-WAIT
- MQGMO-NO-SYNCPOINT
- MQGMO-ACCEPT-TRUNCATED-MSG
- MQGMO-FAIL-IF-QUIESCING

The following Get options are used on the MQGET command when attempting to read messages off a queue in your generated MQ Get command program:

- MQGMO-WAIT
- MQGMO-SYNCPOINT
- MQGMO-ACCEPT-TRUNCATED-MSG

When your MQ Get command program is processing in synchronous mode, MQGMO-NO-SYNCPOINT is used.

In addition in your MQ Get command program, MatchOptions is set to MQMO-MATCH-CORREL-ID. The correlation Id used is the message Id field, if there is a value, in the DFHMAH header of the request message. Otherwise, it is the queue manager generated MsgId on the Put issued in your generated MQ Put command program executed immediately prior to the execution of this program.

User response

Dump the error log (DFHMAERF) to determine the problem. Check to insure the queue manager connection exists and that the queue manager is not quiescing. If the error or warning is reported by a generated MQ Get command program, check the queue attributes and Get wait interval. The wait interval is specified on your type 4 record PARM01 for the program reporting the error in the Properties file update JCL. You can dump the CICS Service Flow Runtime Properties file (DFHMAMPF) using job DFHMAMPD.

You may also want to check the queue to see that it is not GET(DISABLED). Check the REPLYQ specified in the TYPE=2 and TYPE=4 records match, and that the REPLYTOQMGR is specified correctly in the TYPE=2 record. Check that your back-end application is correctly setting the CorrelId.

CIA05005E: MQCLOSE-ERRMSG
Explanation

A call to MQCLOSE has failed

User response

Dump the error log (DFHMAERF) to determine the problem.

Business Transaction Services (BTS) error messages

These error messages are prefaced by **CIA060xx**. You can dump the error log to determine the failed process Id, process-type, program name, transaction, and CICS BTS response codes among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *CICS Business Transaction Services* manual and the *CICS Application Programming Reference* for response codes and definitions.

CIA06001E: DEFINE-PROCESS-ERRMSG

Explanation

An EXEC CICS DEFINE PROCESS has failed in system module DFHMADPL or DFHMADPP when attempting to define a process to the CICS business transaction services system with the CICS Service Flow Runtime Navigation Manager as its root activity.

User response

Dump the error log (DFHMAERF) to determine the problem. 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 found in the DFHMAH header structure is defined and enabled to CICS and that the PROCESS value, if provided in the DFHMAH header structure of the request message, is valid.
- Check that the process name value is not already in use. If it is, there may be an error in your method of allocating a unique process name, or an earlier process using the same name may have failed, and been left in an incomplete state. You can use the CICS Supplied transaction CBAM to see what processes are defined for a process type.

CIA06002E: DEFINE-ACTIVITY-ERRMSG

Explanation

An EXEC CICS DEFINE ACTIVITY has failed in system module DFHMAMGR or in your generated Adapter Navigator program when attempting to define an activity to the CICS business transaction services system. The most likely cause of this error is the target program name or transaction or both are not defined to CICS.

User response

Dump the error log (DFHMAERF) to determine the problem. The BTS error detail specifies 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, insure that all your generated programs and transactions defined in your model to execute for this request name are defined and enabled to CICS. The request name is specified in the DFHMAH header structure of the request message at execution time.

The Navigator program name and transaction Id are specified on your type R record PARM02 and PARM03, respectively, for the desired request name in the Properties file update JCL. You can dump the Properties file (DFHMAMPF) using job DFHMAMPD. The generated command program and FEPI Navigator program names and transactions for the failed request name are properties specified in your

service flow. See the Service Flow Modeler help for information on how these properties are defined.

CIA06003E: RUN-PROCESS-ERRMSG
Explanation

An EXEC CICS RUN ACQPROCESS has failed in the DPL Stub program (DFHMADPL) when attempting to run a process in the CICS business transaction services system with the CICS Service Flow Runtime Navigation Manager as it's root activity. The process is requested to run ASYNCHRONOUS if the request definition indicates *asynchronous mode* and SYNCHRONOUS if the request definition indicates *synchronous mode*.

The request type is specified on your type R record PARM01 for the failed request name in the Properties file update JCL. You can dump the CICS Service Flow Runtime Properties file (DFHMAMPF) using job DFHMAMPD. The request name is found in the DFHMAH header structure of the request message at execution time.

User response

Dump the error log (DFHMAERF) to determine the problem. Check that the request was in *synchronous mode*, and that the transaction and program are available and not defined as remote. There may have been a problem with the repository file - check the CICS log.

CIA06004E: RUN-ACTIVITY-ERRMSG
Explanation

An EXEC CICS RUN ACTIVITY has failed in system module DFHMAMGR or in your generated Adapter Navigator program when attempting to run an activity in the CICS business transaction services system. The activity is requested to run ASYNCHRONOUS if the request definition indicates asynchronous mode and SYNCHRONOUS if the request definition indicates synchronous mode.

The request type is specified on your type R record PARM01 for the failed request name in the Properties file update JCL. You can dump the Properties file, DFHMAMPF, using job DFHMAMPD. The request name is found in the DFHMAH header structure of the request message at execution time.

User response

Dump the error log (DFHMAERF) to determine the problem. 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.

CIA06005E: CHECK-PROCESS-ERRMSG
Explanation

This error is reported by DPL Stub program (DFHMADPL). One of the following conditions exist, the EXEC CICS CHECK ACQPROCESS command has failed, the acquired process was forced to complete, potentially via a CANCEL ACQPROCESS issued outside the CICS Service Flow Runtime, or the Navigation manager (DFHMAMGR) abended.

User response

Dump the error log (DFHMAERF) to determine the problem. Also check the process status using the CBAM transaction (see the *CICS Business Transaction Services* manual). If BTS auditing was turned on for this PROCESSTYPE, you can dump the audit log to help determine the reason for failure. In addition, you should also check the CICS log for diagnostics assistance. If the problem is an abend, check the abend code in the *CICS Messages and Codes* manual for the reason and appropriate actions.

CIA06006E: CHECK-ACTIVITY-ERRMSG
Explanation

This error is reported by Navigation manager (DFHMAMGR) and your generated adapter navigator programs. One of the following conditions exist, the EXEC CICS CHECK ACTIVITY command has failed, the acquired activity was forced to complete, potentially via a CANCEL ACTIVITY issued outside the CICS Service Flow Runtime system, or the activity specified on the command abended.

User response

Dump the error log (DFHMAERF) to determine the problem. Also check the process status using the CBAM transaction (see the *CICS Business Transaction Services* manual). If BTS auditing was turned on for this PROCESSTYPE, you can dump the audit log to help determine the reason for failure. In addition, you should also check the CICS log for diagnostics assistance. If the problem is an abend, check the abend code in the *CICS Messages and Codes* manual for the reason and appropriate actions.

CIA06007E: ACQUIRE-PROCESS-ERRMSG
Explanation

An EXEC CICS ACQUIRE PROCESS command has failed in system module DFHMADPL while attempting to acquire a previously failed CICS Service Flow Runtime process for compensation purposes.

User response

Dump the error log (DFHMAERF) to determine the problem. Check 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. Insure that the process-type is defined and enabled to CICS. Use the CBAM transaction, see *CICS Business Transaction Services* manual for a description, to determine that the process exists and its state.

CIA06011E: CANCEL-PROCESS-ERRMSG
Explanation

An EXEC CICS CANCEL ACQPROCESS command has failed in DPL Stub program (DFHMADPL) while processing in compensation mode and attempting to cancel an acquired previously failed CICS Service Flow Runtime process. The most likely cause of this error is a process mode that is not allowed or one or more of the activities that comprise the process are inaccessible or in CANCELLING mode.

User response

Dump the error log (DFHMAERF) to determine the problem. Use the CBAM transaction, see *CICS Business Transaction Services* manual for a description, to determine the process state.

CIA06017E: RETRIEVE-EVENT-ERRMSG
Explanation

An EXEC CICS RETRIEVE REATTACH EVENT command has failed in the Navigation manager (DFHMAMGR) or in your generated Navigator, FEPI Navigator or server adapters. This condition should not happen when using CICS Service Flow Runtime.

User response

Contact your IBM support center for assistance.

CIA06018E: 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 (DFHMAMGR) or in your generated Navigator, FEPI Navigator or server adapters while processing the defined request flow that might require compensation. An attempt is made to define an input event in the Navigation Manager after recognizing this error and the DEFINE INPUT EVENT command failed. This condition should not happen when using CICS Service Flow Runtime.

User response

Contact your IBM support center for assistance.

CIA06021E INQUIRE-PROCESSTYPE-ERRMSG

Explanation: This error is reported by the CICS SFR Stub 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 flow has not been installed.

System action: None

User response: Check that the specified PROCESSTYPE name is correct and installed in the CICS system. Dump the error log (DFHMAERF) to determine the problem.

CIA06022E ENABLED-PROCESSTYPE-ERRMSG

Explanation: This error is reported by the CICS SFR Stub program (DFHMADPL). The specified PROCESSTYPE resource is not ENABLED in the CICS system.

System action: None

User response: Check that the specified PROCESSTYPE resource is installed and enabled in the CICS system. Dump the error log (DFHMAERF) to determine the problem.

Link3270 server adapter error messages

These error messages are prefaced by **CIA070xx**. You can dump the error log to determine the failed program name, transaction, and any Link3270 bridge header and vector data among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *CICS Application Programming Reference* for CICS response codes and definitions and the *CICS External Interfaces Guide for CICS Version 2 Release 2*, for Link3270 bridge header structure (BRIH) and inbound/outbound vector message structure (BRIV) definitions, return and completion codes.

CIA07001E: DFHL3270-ERRMSG
Explanation

An EXEC CICS LINK command for Link3270 bridge mechanism program, DFHL3270, has failed.

User response

User response: Dump the error log (DFHMAERF) to determine the problem. Check the **CICS resp** and **CICS resp2** fields for the program reporting the error.

CIA07002E: DFHL3270-BRIH-ERRMSG
Explanation

An error was reported by the Link3270 bridge mechanism program, DFHL3270, in the Link3270 bridge header structure, BRIH, fields BRIH-RETURNCODE, BRIH-COMPCODE and BRIH-REASON.

This error message could result from an error in post installation steps, with a Link3270 bridge error (Transaction not found). (BRIH-RETURNCODE = 85).

User response

Dump the error log (DFHMAERF) to determine the problem. Check the following fields in the dump for the program reporting the error:

- Return code
- Comp code
- Reason code

Reference the *CICS External Interfaces Guide Release 2 Version 3* or higher for specific return code values and actions relating to BRIH version 2 errors. Reference the *CICS External Interfaces Guide Version 2 Release 2* for specific return code values and actions relating to BRIH version 1 errors.

If this error indicates a Transaction is not found, an Initiate/Terminate transaction definition required for AOR routing may be defined incorrectly or missing. See field, **Transactionid**, to determine the invalid transaction ID. If the transaction ID indicates CMAI or the transaction ID is equal to the CICS system name (SYSID) where your user transactions (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 165 for Link3270 server adapter AOR routing configuration information.

CIA07010E : 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.

User response

Dump the error log (DFHMAERF). Check the fields **Mapset** and **Map** in the dump for the program reporting the error. Also check the BRIV outbound vector Application Data Structure (ADS) data in the dump and the target CICS application program to determine the problem.

CIA07011W: PROTECTED-UPDATE-WARNING
Explanation

An attempt was made to update a protected map field with data in the BRIV inbound message vector Application Data Structure (ADS). Although this is only a warning, it does end CICS Service Flow Runtime request processing.

User response

Dump the error log (DFHMAERF) to determine the problem. Check the field **Field name** in the dump for the program reporting the error. Modify your adapter server flow for the program reporting the error and redeploy.

CIA07012E: UNEXPECTED-VECTOR-ERRMSG
Explanation

During transaction routing processing, a SEND outbound vector (0404) was not received in a Link3270 server adapter from the CICS Service Flow Runtime Initiate / Terminate transaction (CMAI); program (DFHMALIN).

User response

Dump the error log (DFHMAERF) to determine the problem. Check the field SYSID in the dump for the program reporting the error. Check the CMAI transaction definition in the specified system name of the CICS server region to insure the following:

- That the transaction is defined.
- That the definition indicates the correct CICS Service Flow Runtime Initiate/Terminate program (DFHMALIN).

CIA07013E: NO-VECTOR-ERRMSG
Explanation

No BRIV outbound vector Application Data Structure (ADS) data was received in a Link3270 server adapter when one was expected.

User response

Dump the error log (DFHMAERF) to determine the problem. Check the Link3270 bridge header structure, **BRIH**, and **BRIV** outbound vector header control fields to determine what was sent. Check, and possibly modify your adapter server flow for the program and service name (optional) specified and redeploy. Also, check the target CICS application program to determine possible processing problems or unsupported functions.

CIA07014E: INQUIRE-TRANSID-ERRMSG
Explanation

An EXEC CICS INQUIRE TRANSACTION command has failed. The most likely cause is a transaction is not defined in the **CICS CSD**.

User response

Dump the error log (DFHMAERF) to determine the problem. Check the following fields in the dump for the program reporting the error:

- Transid
- CICS Resp
- CICS Resp2

Check to ensure the transaction identified in the dump is defined to **CICS CSD**. See the *CICS Resource Definition Guide*.

CIA07015W: INVALID-ATTRIBUTE-WARNING
Explanation

An invalid attribute was found in the BRIV outbound vector Application Data Structure (ADS) data. This is only a warning. The attribute value is set to LOW-VALUES. The most likely cause is a map definition error or target CICS application program error.

User response

Dump the error log (DFHMAERF) to determine the problem. Check the following fields in the dump for the program reporting the error to determine the error:

- Field name
- Invalid attribute

Check to ensure the map and map field definitions are valid and supported and the target CICS application program for processing errors or unsupported functions.

CIA07016E: MAPSET-LOAD-ERRMSG
Explanation

An EXEC CICS LOAD PROGRAM command for the specified map set load module has failed. The most likely cause is map set load module is not defined in the **CICS CSD** and/or the map set load module could not be found in the **CICS RPL** concatenation.

User response

Dump the Error log (DFHMAERF) to determine the problem. Check the field **Mapset** in the dump for the program reporting the error. Update the **CICS CSD** and/or **CICS RPL** concatenation as necessary to define and/or include the map set load module, respectively, in the CICS Service Flow Runtime router region.

CIA07017E: 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.

User response

Dump the Error log (DFHMAERF). Check the fields **Mapset** and **Map** in the dump for the program reporting the error. Check the **CICS 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 to insure the correct map set load module is defined to the CICS Service Flow Runtime router region.

CIA07901E: UNSUPPORTED-VECTOR-ERRMSG
Explanation

One of the following unsupported BRIV outbound vectors was received in the CICS Service Flow Runtime Link3270 Vector Processing module, DFHMAVCP, in non-passthrough mode.

- SEND (0404)
- SEND TEXT MAPPED (1806)
- SEND TEXT NOEDIT (1806)
- SEND MAP with ACCUM (0406)

User response

These **BRIV** outbound vectors are unsupported in this release. These unsupported vectors are CICS SFR limitations to functions not supported by the Link3270 mechanism itself. See “Use restrictions for Link3270 bridge mechanism” on page 174 for information on additional run time restrictions.

Dump the Error log (DFHMAERF) to determine the business request adapter server flow. Modify your adapter server flow for the program specified and if possible redeploy as a FEPI server adapter.

CIA07999E: UNSUPPORTED-CICS-ERRMSG
Explanation

The release of CICS does not support the Link3270 bridge mechanism

User response

Upgrade to supported **CICS Transaction Server Version 2 Release 2** or later. Dump the error log (DFHMAERF) to determine the business request adapter server flow. Modify your adapter server flow for the program specified and if possible redeploy as a FEPI server adapter.

Additional error messages

These error messages are prefaced by **CIA080xx**, with the exception of abend messages which are prefaced by **CIA99999**. You can dump the error log to determine the failed program name, transaction and CICS response codes among other pertinent information. Use this information to assist in problem diagnosis.

Abend messages

At run time, the system normally intercepts abends by including an active **EXEC CICS HANDLE ABEND** command. The actual 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 CIA99999.

CIA08001E: EIBCALEN-ERRMSG
Explanation

An attempt was made to execute the system module DFHMADPL or DFHMADPP from a client application or service requestor with an EIBCALEN equal to zero.

User response

Fix the client application or service requestor to pass the correct structure into the Stub program (DFHMADPL) or (DFHMADPP) . The CICS Service Flow Runtime DPL Stub program (DFHMADPL), expects the DFHMAH header structure followed by the client application or service requestor request data. The CICS SFR Passthrough Stub program (DFHMADPP), expects the DFHMAH header structure followed by the DFHMAH2 header structure followed by the client application or service requestor request data.

CIA08002E: COMMAREA-ERRMSG
Explanation

The length of the application request message as specified in field, DFHMAH-DATALENGTH, passed in the CICS Service Flow Runtime message header structure, DFHMAH, was greater than the length of the mapped input request area in the identified adapter Navigator program; or the value of DFHMAH-DATALENGTH + DFHMAH-STRUCLENGTH exceeded EIBCALEN in the DPL Stub program (DFHMADPL); or the value of DFHMAH2-DATALENGTH + DFHMAH-STRUCLENGTH + DFHMAH2-STRUCLENGTH exceeded EIBCALEN in the Passthrough Stub program (DFHMADPP).

See “Request message headers” on page 80 for a detailed description of the request message.

User response

Fix the service requestor to pass the correct application request message length or correct the modeled adapter flow. The length passed in the message header DFHMAH is compared to the LENGTH OF INPUT-DATA-CONTAINER in the adapter Navigator program. The INPUT-DATA-CONTAINER identifier is a Cobol 1 group item that has elementary items in the generated copybook name found in the COPY statement subordinate to the 01-level of INPUT-DATA-CONTAINER. The DFHMAH-DATALENGTH must not be greater than the sum of the lengths, in bytes, of the elementary items found in this copybook.

CIA08003E: TDQ-WRITE-ERRMSG
Explanation

A WRITEQ TD command has failed in the CICS Service Flow Runtime to the intrapartition transient data error queue, CMAQ. The most likely cause is that the transient data queue is DISABLED or no more space exists on the queue. The error is written to the console. For information on how TDQ errors are written to the console, see “Conditions that cause CICS Service Flow Runtime to write to the system console” on page 197

User response

Check that the transient data queue, CMAQ, is not DISABLED. If the NOSPACE condition is encountered, ensure that the transient data error queue (CMAQ), **CICS**

CSD definition is correct. The default installed TRANSID and TRIGGERLEVEL attribute value settings for the CMAQ transient queue definition are CMAQ and +1, respectively. Also, check that the CICS SFR error logging transaction CMAQ is defined and ENABLED.

CIA08004E: TDQ-READ-ERRMSG
Explanation

A **READQ TD** command has failed in the CICS Service Flow Runtime error logging transaction CMAQ, to the intrapartition transient error queue CMAQ, with a condition other than ZERO. The most likely cause is that the transient data queue is DISABLED. The error is written to the console. For information on how TDQ errors are written to the console, see “Conditions that cause CICS Service Flow Runtime to write to the system console” on page 197

User response

Check to insure that the transient data queue CMAQ is not DISABLED.

CIA08005E: VALIDATION-ERRMSG
Explanation

An invalid field value was found in the CICS SFR passthrough header structure DFHMAH2.

User response

Dump the error log (DFHMAERF) to determine the cause of the problem. Also see “DFHMAH2 header structure” on page 87 to determine field names and valid values.

CIA08006E: CICS-LEVEL-ERRMSG
Explanation

The CICS Service Flow Runtime router CICS region is not at CICS Transaction Server for z/OS V3.1 or higher.

User response

Upgrade the version of CICS to the necessary prerequisite

CIA08007E: INVALID-FORMAT-ERRMSG
Explanation

The request message coming from the service requestor into the CICS Service Flow Runtime is in an invalid format. This error message can occur for the following reasons:

- An inbound request message was received in the CICS SFR DPL Stub program (DFHMADPL), indicating a passthrough format. This can occur if the following conditions exist:
  - the CICS SFR header structure DFHMAH indicates a version +2 header structure
  - the field DFHMAH-VERSION equal +2, and the field DFHMAH-FORMAT equal 'MAH2,' indicating passthrough
  - the field DFHMAH-UOWCONTROL equal +3, indicating passthrough.

The existence of the passthrough header structure DFHMAH2, without the above field settings will not result in this error condition, but might cause CICS SFR adapter flow processing failure, as all request data subsequent to the DFHMAH header structure will be treated as application input data.

- A request message from the service requestor was received in the CICS SFR Passthrough Stub program (DFHMADPP), that does not indicate a passthrough request message format. The header structure DFHMAH must indicate the following:
  - a version +2 header structure, field DFHMAH-VERSION equal +2
  - a Link3270 passthrough format, field DFHMAH-FORMAT equal 'MAH2'
  - a CICS SFR Passthrough processing, field DFHMAH-UOWCONTROL equal +3.
- An invalid CICS SFR Passthrough request message from the service requestor was received in the Link3270 Passthrough program (DFHMALPT), transaction ID = 'CMAL'.

User response

Check that the correct CICS SFR request message format and header structure field settings are used in the client application.

In the case of an invalid CICS SFR Link3270 Passthrough request message received in the program (DFHMALPT), the most likely cause is a missing CIA-SCREEN-HEADER or CIA-MAP-HEADER. The screen header and map header must immediately follow the DFHMAH and DFHMAH2 header structures in the inbound passthrough request message. See “DFHMAH2 header structure” on page 87.

CIA08008E 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 Modeler 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.

CIA08010E DATA-LENGTH-ERRMSG

Explanation: The length of data in a channel container passed to DFHMADPL was outside the allowed range.

System action: Execution of the CICS Service Flow Runtime component is terminated. This error can occur in the following CICS SFR programs: DFHMADPL

User response: Correct the client application or service requestor to pass the correct channel container contents into DFHMADPL.

CIA08011E INPUT-COMBINATION-ERRMSG

Explanation: An incorrect combination of channel containers was passed to DFHMADPL.

System action: The DFHMADPL server adapter terminated.

User response: Correct the client application or service requestor to pass the correct channel container combinations into DFHMADPL.

CIA08009E: POST-ERR-ERRMSG

CIA08009 Error, *error msg*, reported in CICS SFR Transaction(*transid*), Program (*prog*) Applid (*applid*); Unable to post error.

Explanation

An error was encountered within the CICS SFR request. However, CICS SFR was unable to post the error. The most likely cause is that the CICS transient data error queue (CMAQ) was not defined, is not available or has run out of storage. If WMQ is being used to post error messages, then there could be a problem with the WMQ setup. The values are:

error msg
: Error message id that CICS SFR was trying to report.

transid Transaction id of the program reporting the error.

prog Program trying to report the error.

applid APPLID id of the system reporting the error.

Note: This message is an operator message only.

User response

Check to ensure that the transient data queue is defined, enabled and has space. If WMQ is being used for posting errors, check that the queue, CIA.SYSTEM.ERROR.QUEUE, is defined correctly.

API error messages

These error messages are prefaced by **CIA081xx**, and result from failed CICS API commands. You can dump the error log to determine the failed program name, transaction and CICS response codes among other pertinent information. Use this information to assist in problem diagnosis.

Reference the *CICS Application Programming 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.

CIA08101E: CICS-RECEIVE-ERRMSG

Explanation

An **EXEC CICS RECEIVE** command failed in the CICS Service Flow Runtime. This error can occur in the following CICS SFR programs:

- DFHMALSC

  DFHMALSC is the CICS SFR Link3270 State Cleanup module for adapter services of the single connector nonpersistent type, where state information is stored, retrieved and deleted in temporary storage queues. See “Managing state information” on page 179 for processing descriptions of State Cleanup programs.
- DFHMALFC

  DFHMALFC is the CICS SFR Link3270 State Cleanup program for adapter services of the single connector persistent type and for aggregate deployment patterns, where state information is stored, retrieved and deleted in the Link3270 VSAM State file DFHMAL2F. See “Managing state information” on page 179 for processing descriptions of State Cleanup programs.

User response

The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes.

Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08102E: 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 the CICS SFR Link3270 State Cleanup program for adapter services of the single connector nonpersistent type, where state information is stored, retrieved and deleted in temporary storage queues.
- DFHMALFC

  DFHMALFC is the CICS SFR State Cleanup program for adapter services of the single connector persistent type and of adapter services of the aggregate connector deployment types, where state information is stored, retrieved and deleted in the Link3270 VSAM State file (DFHMAL2F).
- DFHMALFD

  DFHMALFD is the CICS SFR Link3270 Facility Deallocate program that is initiated as the result of a **CICS START** command issued from the following CICS SFR programs:
  - DFHMALSC
  - DFHMALFC.

See “Managing state information” on page 179 for processing descriptions of the State Cleanup programs.

User response

The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes.

Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08103E: 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 the CICS SFR Link3270 State Cleanup module for adapter services of the single connector nonpersistent types, where state information is stored, retrieved and deleted in temporary storage queues.

- DFHMALFC

  DFHMALFC is the CICS SFR Link3270 State Cleanup program for adapter services of the single connector persistent types and of aggregate connector persistent and nonpersistent deployment types 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 of using their transaction Ids with an INTERVAL as defined in the terminal data when initially started via terminal input or as defined in the **INITPARM** parameter in the CICS SIT for the CICS SFR router region. DFHMALSC transaction ID is CMAK. DFHMALFC transaction ID is CMAF. Each module can also schedule STARTs of transaction CMAD, when CICS SFR Link3270 facilities may need to be de-allocated.

User response

The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes.

Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08104E: CICS-ASSIGN-ERRMSG
Explanation

An **EXEC CICS ASSIGN** command failed in the CICS Service Flow Runtime.

User response

The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes.

Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08106E: CICS-ENQUEUE-ERRMSG
Explanation

An **EXEC CICS ENQ** command failed in the CICS Service Flow Runtime.

This error can occur in the following CICS Service Flow Runtime programs:

- DFHMALSC

  DFHMALSC is the Link3270 State Cleanup module for adapter services of the single connector nonpersistent type, where state information is stored, retrieved and deleted in temporary storage queues.

- DFHMALTS

DFHMALTS is the Link3270 State information storage, retrieval and delete program for adapter services of the single connector nonpersistent type, where state information is stored, retrieved and deleted in CICS SFR Link3270 temporary storage queues.

User response

The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes.

Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08107E: CICS-DEQUEUE-ERRMSG
Explanation

An **EXEC CICS DEQ** command failed in the CICS Service Flow Runtime. This error can occur in the following CICS SFR programs:

- DFHMALTS

  DFHMALTS is the Link3270 State information storage, retrieval and delete module for adapter services of the single connector nonpersistent type, where state information is stored, retrieved and deleted in CICS Service Flow Runtime Link3270 temporary storage queues.
- DFHMALSC

  DFHMALSC is the Link3270 State Cleanup program for adapter services of the single connector nonpersistent type, where state information is stored, retrieved and deleted in temporary storage queues.

User response

The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes.

Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08108E : CICS-INQUIRE-ERRMSG
Explanation

An **EXEC CICS INQUIRE** command failed in the CICS Service Flow Runtime.

User response

The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes.

Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API

commands and associated CICS response code values, meaning and appropriate user actions.

CIA08109E CICS-GETMAIN-ERRMSG

Explanation: An **EXEC CICS GETMAIN** command failed in the CICS Service Flow Runtime.

System action: The DFHMADPL server adapter terminated.

User response: The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes. Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08110E CICS-FREEMAIN-ERRMSG

Explanation: An **EXEC CICS FREEMAIN** command failed in the CICS Service Flow Runtime.

System action: The DFHMADPL server adapter terminated.

User response: The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes. Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08111E CICS-SOAPFAULT-ERRMSG

Explanation: An **EXEC CICS SOAPFAULT CREATE** command failed in the CICS Service Flow Runtime.

System action: The DFHMADPL server adapter terminated.

User response: The CICS Service Flow Runtime error log specifies the program module and transaction in which the CICS API command failed and the associated CICS response codes. Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the specific CICS API commands and associated CICS response code values, meaning and appropriate user actions.

CIA08112E CICS-INVOKEWS-ERRMSG

Explanation: An **EXEC CICS INVOKE WEBSERVICE** command failed in the CICS Service Flow Runtime.

System action: The CICS Service Flow Runtime error log specifies the associated CICS response codes for the failure with the CICS API command.

User response: Dump the error log (DFHMAERF) to determine the cause of these problems. Reference the *CICS Application Programming Reference* for the **EXEC CICS INVOKE WEBSERVICE** API command and associated CICS response code values, meaning, and appropriate user actions.

CIA082*xx* update properties file error messages

When you run the job to update the properties file, the following messages can be issued in the job output.

CIA08201I Overwrite mode enabled.

Explanation: **MODE**=OVERWRITE has been specified in the update properties file job.

System action: Processing continues.

User response: None

CIA08202I Safe mode enabled.

Explanation: **MODE**=SAFE has been specified in the update properties file job.

System action: Processing continues.

User response: None.

CIA08203W Safe mode assumed.

Explanation: **MODE**=*value* value was incorrectly specified in the update properties file job. Only SAFE or OVERWRITE are allowed as values for this parameter.

System action: Safe mode is selected as the default mode.

User response: If you do not want to run the update properties file job in safe mode, specify MODE=OVERWRITE. This will overwrite any existing adapter service definitions.

CIA08204W Mode indicator not found

Explanation: The **MODE** parameter was not specified in the update properties file job.

System action: Safe mode is assumed.

User response: If you do not want to run the update properties file job in safe mode, specify MODE=OVERWRITE. This will overwrite any existing adapter service definitions.

CIA08211I Overwrite mode started

Explanation: Overwrite mode has begun.

System action: Property file updates will occur.

User response: None.

CIA08212I Safe mode checking started.

Explanation: Safe mode checking has started.

System action: The properties file update job will check to see if there are any record name clashes in the properties file.

User response: None.

CIA08213I Safe mode checking complete.

Explanation: Safe mode checking has completed successfully. No name clashes were found in the properties file.

System action: The properties file will now be updated.

User response: None.

CIA08214I Updating properties file.

Explanation: Safe mode has finished checking and is now updating the properties file.

System action: The properties file is being updated.

User response: None.

CIA08215E Safe mode found existing records. No updates performed.

Explanation: Name clashes were found in the properties file during the safe mode checking.

System action: No updates were made to the properties file.

User response: See message CIA08216I for further help.

CIA08216I Specify `MODE=`OVERWRITE to force updates.

Explanation: See message CIA08215E.

System action: None.

User response: If you want to force updates to the properties file to override the existing definitions, use `MODE=OVERWRITE`.

CIA08217I Properties file update complete.

Explanation: All updates have been made to the properties file.

System action: None.

User response: None.

CIA08221E Record *record_name* **already exists in the properties file.**

Explanation: Existing records with the same name have been found during the safe mode checking.

System action: No updates will be made to the properties file.

User response: To force updates to occur, run the update properties file specifying `MODE=OVERWRITE`.

CIA08222W Overwriting record *record_name***.**

Explanation: Record *record_name* already exists in the Properties files.

System action: This record has been updated because `MODE=OVERWRITE` was specified on the properties file update job.

User response: None.

XML parsing error messages

These error messages are prefaced by **CIA083xx**. You can dump the error log to determine the failed program name, transaction and XML exception codes among other pertinent information. Use this information to assist in problem diagnosis.

The following error messages are issued as the result of failed XML parse or build processing. Reference the *Enterprise COBOL for z/OS and OS/390 Programming Guide* for any XML exception codes reported.

For a sample of XML message structures for both non-passthrough and passthrough modes, see "XML message formats" on page 322 in the Samples section.

CIA08301E: 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 due to some other unexpected system event. This error condition can be raised in the following CICS SFR programs:

- DFHMXMI

  DFHMAXMI is called by the DPL Stub program (DFHMADPL) and the Passthrough Stub program (DFHMADPP). DFHMAXMI parses the inbound XML request message and returns the appropriate CICS SFR header structure, DFHMAH and possibly DFHMAH2, in fixed format to the calling system program.
- DFHMAXMO

  DFHMAXMO is called by the following CICS SFR programs:
  - DPL Stub program DFHMADPL
  - Passthrough Stub program DFHMADPP
  - Navigation Manager DFHMAMGR

  DFHMAXMO builds the outbound XML reply message using the appropriate CICS SFR header structure, DFHMAH and possibly DFHMAH2, and any application reply data passed to it from the calling system module.
- DFHMAXPI

  DFHMAXPI is called by the CICS SFR Passthrough Stub program (DFHMADPP). DFHMAXPI parses the inbound XML passthrough request message (the portion of the XML request message following the CICS SFR header structures, DFHMAH and DFHMAH2, and returns the passthrough application data in fixed format to the calling system program.

  DFHMAXPI also builds the outbound XML passthrough application reply message using any passthrough application reply data passed to it from the calling system program.

User response

Dump the Error log (DFHMAERF) to determine the problem.

Check any XML exception code reported. Reference the *Enterprise COBOL for z/OS and OS/390 Programming Guide* for XML exception codes and definitions. Also, examine destination **CEEMSG** for messages output by language environment to determine why the XML parse failed.

CIA08302E: XML-NO-SCREEN-ERRMSG
Explanation

A 'screen tag' was not found in the CICS Service Flow Runtime Link3270 Passthrough XML program (DFHMAXPI) when parsing the XML document.

The most likely cause is a missing passthrough screen header structure, CIA-SCREEN-HEADER, in the inbound request message.

The screen header structure is mapped by copybook, DFHMALSH. DFHMAXPI is called by the CICS SFR Passthrough Stub program (DFHMADPP).

User response

Check that the correct Passthrough request message format and header structure field settings are used in the client application or service requestor. The screen header and map header must immediately follow the DFHMAH and DFHMAH2 header structures in the inbound passthrough request message.

For a sample of XML message structures for both non-passthrough and passthrough modes, see “XML message formats” on page 322 in the Samples section.

See “DFHMAH2 header structure” on page 87 for information on the fields in the passthrough message header DFHMAH2.

CIA08303E: XML-NO-MAP-ERRMSG
Explanation

A 'map' tag was not found in the CICS Service Flow Runtime Link3270 Passthrough XML program (DFHMAXPI) when parsing the XML document.

The most likely cause is a missing passthrough map header structure, CIA-MAP-HEADER, in the inbound request message.

The map header structure is mapped by copybook, DFHMALMH.

DFHMAXPI is called by the CICS SFR Passthrough Stub program (DFHMADPP).

User response

Check that the correct Link3270 passthrough request message format and header structure field settings are used in the client application or service requestor.

The screen header and map header must immediately follow the DFHMAH and DFHMAH2 header structures in the inbound passthrough request message.

For a sample of XML message structures for both non-passthrough and passthrough modes, see “XML message formats” on page 322 in the Samples section.

See “DFHMAH2 header structure” on page 87 for information on the fields in the passthrough message header DFHMAH2.

CIA08304E: XML-NO-FIELD-ERRMSG
Explanation

A 'field' tag was not found in the CICS Service Flow Runtime Link3270 Passthrough XML program (DFHMAXPI) when parsing the XML document.

The most likely cause is an invalid passthrough request message format.

User response

Check that the correct Link3270 Passthrough request message format and header structure field settings are used in the client application or service requestor.

The screen header and map header must immediately follow the DFHMAH and DFHMAH2 header structures in the inbound passthrough request message.

For a sample of XML message structures for both non-passthrough and passthrough modes, see “XML message formats” on page 322 in the Samples section.

See “DFHMAH2 header structure” on page 87 for information on the fields in the passthrough message header DFHMAH2.

CIA08305E: XML-INVALID-TAG-ERRMSG
Explanation

An invalid or missing tag was found in the CICS Service Flow Runtime Link3270 Passthrough XML program (DFHMAXPI) when parsing the XML document.

The most likely cause is an invalid passthrough request message format.

User response

Check that the correct Link3270 Passthrough request message format and header structure field settings are used in the client application or service requestor.

The XML passthrough request message should have tag names of 'screen', 'map', 'field', 'text', and 'input'.

The screen header and map header must immediately follow the DFHMAH and DFHMAH2 header structures in the inbound passthrough request message.

For a sample of XML message structures for both non-passthrough and passthrough modes, see “XML message formats” on page 322 in the Samples section.

See “DFHMAH2 header structure” on page 87 for information on the fields in the passthrough message header DFHMAH2.

CIA08306E : XML-FIELD-LEN-ERRMSG
Explanation

Either an invalid field length for an ADS field passed in the CICS Service Flow Runtime XML request message or an invalid BMS OCCURS attribute length was found in the CICS SFR Link3270 Passthrough XML program (DFHMAXPI) when parsing the XML document.

The most likely cause is an invalid passthrough request message format.

User response

Check that the correct Link3270 passthrough request message format and header structure field settings are used in the client application or service requestor. An 'oi' or 'OI' tag was found but the length of the attribute was greater than 4 bytes in length.

The IVP messages

The following messages are issued by the IVP.

CIA09000

CIA09000: The CICS SFR has started.

Explanation

This message confirms that the IVP has been started by the CMAA transaction.

CIA09001

CIA09001: The CICS SFR has finished

Explanation

This message is issued when the IVP has completed. Check the log for other messages before this message was issued to determine if any errors occurred when the IVP was validating the selected adapters.

CIA09002

CIA09002: The CICS SFR *adapter_short_name* Server Adapter completed successfully

Explanation

This message confirms that the IVP successfully validated the set up for a particular server adapter, where *adapter_short_name* could be one of the following values:

- 3270LINK BRIDGE
- FEPI
- DPL
- WMQ

CIA09003

CIA09003: The CICS SFR *adapter_short_name* Server Adapter failed.

Explanation

This message indicates that the IVP was unable to successfully validate the set up for a particular server adapter, where *adapter_short_name* could be one of the following values:

- 3270LINK BRIDGE
- FEPI
- DPL
- WMQ

User response

Check the TDQ CSMT for other error messages to find out the specific failure that occurred when the IVP was attempting to validate the adapter.

CIA09004

CIA09004: Adapter returned data in error

Explanation

The IVP failed for the adapter specified in message CIA09003 because the data that was returned to the IVP by CICS does not match the expected value.

User response

Look in the CICS job output for any error messages that might be related to the setup of CICS SFR in the region.

Dump the Error file DFHMAERF and look for any relevant messages. If you are using Link3270, also dump the Vector log file DFHMALVF using the provided utility.

Run CICS trace to see if this highlights any errors.

If you can not find relevant messages or errors, contact IBM Support.

CIA09005

CIA09005: CICS API error - request *cics_request* resp: *value* resp2: *value*

Explanation

A CICS API request has failed, where

cics_request
is one of the following commands:

- LINK PROGRAM
- RECEIVE MAP
- RETURN TRANSID
- SEND MAP
- SEND TEXT

and *value* is the response code issued by CICS.

User response

Look in the *CICS Application Programming Reference* for an explanation of the returned response codes that were issued for the appropriate command.

CIA09006

CIA09006: CICS INQUIRE FILE (*file_name*) failed - resp: *value* resp2: *value*

Explanation

The IVP was unable to inquire on the specified file for the selected adapter, where *file_name* can be one of the following:

- DFHMABCF

- BTS
- DFHMAERF
- DFHMAMPF
- DFHMALRF
- DFHMALVF
- DFHMAL2F
- DFHMACOF
- DFHMATIF

and *value* is the response code issued by CICS.

User response

Check that the file has been correctly defined and installed in the CICS region. Look in the *CICS Application Programming Reference* for an explanation of the returned resp and resp2 values.

Abend error messages

CIA99999E: ABEND-ERRMSG

Explanation

An abend has occurred in a CICS Service Flow Runtime programs or one of your generated programs.

User response

Dump the error log (DFHMAERF) to determine the abend code. Check the abend code in the *CICS Messages and Codes* manual for the cause and appropriate actions.

CIAI*xxxx* error messages

These messages are issued when verifying the post-installation procedure DFHMAINJ.

CIAI1000I Validation of input parameters is taking place.

Explanation: Validation of user-supplied input data is taking place.

System action: Job continues.

User response: None.

CIAI1001I 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.

CIAI1002I 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.

CIAI1003E 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.

CIAI1004E 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.

CIAI1005E 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.

CIAI1006E Allocate failed for `CSDNAME` = *csdname*

Explanation: The CICS DFHCSD file *csdname* could not 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.

CIAI1007E 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.

CIAI1008E 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.

CIAI1009E 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.

CIAI1010I Customizing member: *membername*

Explanation: Customization has begun for the member *membername*

System action: Job continues

User response: None

CIAI1011I SCIZSAMP customization ended without errors

Explanation: Customization has completed successfully

System action: Job continues

User response: None

CIAI1012I SCIZSAMP customization ended with errors

Explanation: Customization has failed. Previous messages describe the errors.

System action: Job terminates.

User response: Correct the errors found in any previous messages and rerun the job.

CIAI1013E No continuation line found for name

Explanation: The continuation character (\) was used in the previous line but the next non-comment line does not have the continuation character as the first non-blank character.

System action: Job fails.

User response: Check your input and rerun DFHMAINJ. If the problem still persists, contact IBM.

CIAI1014S DFHMAINR has encountered an unrecoverable error. Novalue error at line *lineno* of DFHMAINR: *errtxt*

Explanation: The REXX exec DFHMAINR has encountered an unrecoverable error.

System action: Job fails

User response: Check your input and rerun DFHMAINJ. If the problem still persists, contact IBM.

CIAI1015S DFHMAINR has encountered an unrecoverable error. REXX error *rc* at line *lineno* of DFHMAINR: *errtxt*

Explanation: The REXX exec DFHMAINR has encountered an unrecoverable error.

System action: Job fails

User response: Check your input and rerun

DFHMAINJ. If the problem still persists, contact IBM.

CIAI1016S DFHMAINR has encountered an unrecoverable error. Error *err* **at line** *lineno* **of DFHMAINR:** *errtxt*

Explanation: The REXX exec DFHMAINR has encountered an unrecoverable error.

System action: Job fails.

User response: Check your input and rerun DFHMAINJ. If the problem still persists, contact IBM.

CIAI1017E 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.

CIAI1018E If MQ=YES is specified then you must specify HLQMQ

Explanation: When **MQ**=YES is specified then you are required to specify **HLQMQ**.

System action: Job fails

User response: Specify the **HLQMQ** parameter and value in your input and rerun DFHMAINJ.

CIAI1019E Validation failed for *parameter = value***.**

Explanation: Validation of user-supplied parameter *parameter=value* has failed.

System action: Job fails.

User response: See following CIAI1020E message for an explanation of why the error occurred.

CIAI1020E *Problem data from CIAI1019E***.**

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.

Abends

These abends are issued by CICS SFR.

CIAX
Explanation

While processing in synchronous rollback mode, an error or failure occurred in adapter request processing.

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 DPL Stub (DFHMADPL) reports the abend to the error file DFHMAERF, as a RUN-PROCESS-ERRMSG. See “Business Transaction Services (BTS) error messages” on page 233 for information on the run process error message CIA06003.

User response

Dump the error log to determine the problem in adapter request processing. See Chapter 10, “Troubleshooting and support,” on page 199 for information on how to dump the error file.

CIAY
Explanation

When running the CICS SFR IVP, an abnormal response was received from one of the following:

- EXEC CICS RETURN TRANSID
- EXEC CICS SEND TEXT
- EXEC END MAP

System action

Following this failure, the transaction performs the following actions:

- Issues the message CIA09005 (see “CIA09005” on page 254), which provides the details of the request to CICS as well as the response.
- Creates an exception trace, where:

  tracenum=100
  This is '0064'x in hexidecimal

  resource='CSFRIV*nn*'
  nn can have the values of 49 through 58, providing a unique identification of the failure detection point

  from=*program_data*
  program_data is the IVP application program data that was returned.
- Issues an ABEND (CIAY) request, that rolls back any updates.

User response

Determine the precise failure from the CICS SFR messages and exception tracing. You might need to rerun the IVP for the adapter that failed with auxiliary tracing switched on for user programs. Look for other CICS messages that might help to explain the error condition.

If you can not correct the error or find its cause, contact IBM support.

Applying APARs

When you need to apply maintenance or fixes to the runtime environment, use the following process.

Maintenance is applied using the a JCL batch job called DFHMAINA. The JCL is very similar to the DFHMAINJ post-installation 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.

1. Edit the DFHMAINA JCL, supplying the same parameter values as you would for DFHMAINJ. The parameters that you need to edit are:

   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, as it 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**.

MQ YES|NO
Indicates if you require support for WebSphere MQSeries (WMQ). If you specify `MQ YES`, then you must also define values for the **HLQMQ** and **THE_QMGR** parameters. You must specify this parameter if you want to use the Simulator program to verify your installation.

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 HFS 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 should not exceed 255 characters, and should 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 HFS. 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 should not exceed 255 characters, and should 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 HFS that contains subdirectories for the requester mode pipeline configuration files and Web service requester binding files. The length of the fully qualified directory name should not exceed 255 characters, and should 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 HFS 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 should not exceed 255 characters, and should 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 HFS. 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 should not exceed 255 characters, and should 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 HFS that contains subdirectories for the provider mode pipeline configuration files and Web service provider binding files. The length of the fully qualified name should not exceed 255 characters, and should 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, this would rename every jobname in the members of the runtime SCIZSAMP library with CSFR as the first four characters e.g. //DFHMASET would become //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.

HLQMQ *your.mq.hlq*
A 1-35 character length value that is the high level qualifier of the WMQ runtime libraries.

Note: This parameter is required if you specified `MQ YES`.

THE_QMGR *qmgrname*
A 1-4 character length value that is the name of the WMQ queue manager that resources are defined to.

Note: This parameter is required if you specified `MQ YES`.

Acceptable characters:
`A-Z a-z 0-9`

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

2. Submit DFHMAINA. The job is validated by default. This is recommended, as any errors in the parameter values are highlighted immediately before the job runs. DFHMAINA invokes the REXX program DFHMAINX. This 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 should have a `Customizing member` message associated with it in the output. You should also see the following messages in the //SYSTSPRT of the DFHMAINA job output:

   ```
   CICS SFR SCIZSAMP customization beginning
   CICS SFR SCIZSAMP customization completed succesfully
   ```

   If you have APAR PK32131 applied, the following messages are included in the output:

   ```
   CIAI1002I SCIZSAMP customization beginning
   CIAI1011I 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 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 need to reapply the fix. If the problem still persists, contact IBM.
4. Perform any additional steps as outlined in the APAR documentation.

Part 2. Samples

The following samples are provided to help you with various aspects of administering the runtime environment.

© Copyright IBM Corp. 2001, 2010

Chapter 11. Samples

JCL

The following sample JCL is provided with the CICS Service Flow Runtime.

Properties file update JCL (DFHMAMPU)

```
//jobname  JOB DFHMAMPU,'DFHMAMPU',CLASS=A,MSGCLASS=H TYPRUN=SCAN
//*******************************************************************
//* @START_RRS_COPYRIGHT@
//* Licensed Materials - Property of IBM
//*
//* "Restricted Materials of IBM"
//*
//* 5655-M15
//*
//* (c) Copyright IBM Corp. 2000, 2006
//*
//* CICS Service Flow Runtime
//* @END_RRS_COPYRIGHT@
//*******************************************************************
//*******************************************************************
//* RUN DFHMAMUP (CICS SFR PROPERTIES FILE UPDATE PROGRAM)
//*
//*******************************************************************
//MAMUP    EXEC PGM=DFHMAMUP
//STEPLIB DD DSN='hlq.SCIZLOAD',DISP=SHR
//*******************************************************************
//*     The output will go to SYSPRINT
//*******************************************************************
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//DFHMAMPF DD DSN=hlq.DFHMAMPF,DISP=SHR
//*************************************************************************
//*                                                                       *
//*   SYSIN control card formats                                          *
//*                                                                       *
//*   Card=MODE  ********************************************************
//*                                                                       *
//*  MODE (SAFE; OVERWRITE)                                               *
//*                                                                       *
//**  Card=TYPE  ********************************************************
//*                                                                       *
//*  TYPE (R-REQUEST; 0-NAVIGATOR; 1-DPL; 2-MQPUT; 3-FEPI; 4-MQGET;       *
//*                  5-LINK3270)                                          *
//*                                                                       *
//**  Card=NAME  ********************************************************
//*                                                                       *
//*  NAME (REQUEST NAME/COMMAND NAME)                                     *
//*                                                                       *
//*                                                                       *
//**  PARMS TYPE=R  *****************************************************
//*                                                                       *
//*  PARM01      : REQUEST TYPE (0-ASYNC; 1-SYNC; 2-SYNC ROLLBACK)        *
//*  PARM02      : INITIAL PROGRAM ID                                     *
//*  PARM03      : INITIAL TRANSACTION ID                                 *
//*  PARM04      : PERSISTENCE IND                                        *
//*  PARM05      : XML PARSE PROGRAM ID                                   *
//*  PARM06      : DEPLOYMENT IND                                         *
//*  PARM07      : INITIAL PROGRAM TYPE                                   *
//*                                                                       *
//**  PARMS TYPE=0  *****************************************************
//*                                                                       *
//*  (reserved)                                                           *
```

© Copyright IBM Corp. 2001, 2010

```
//*                                                                    *
//**  PARMS TYPE=1  ***************************************************
//*                                                                    *
//*  PARM01      : DPL PROGRAM                                         *
//*  PARM02      : DPL SYSID                                           *
//*  PARM03      : DPL MIRROR TRANSACTION                              *
//*  PARM04      : DPL SYNCONRETURN (Y/N)                              *
//*                                                                    *
//**  PARMS TYPE=2  ***************************************************
//*                                                                    *
//*  PARM01      : MQ MSGTYPE  (1-REQUEST; 2-REPLY; 8-DATAGRAM)        *
//*  PARM02      : MQ REQUEST QNAME                                    *
//*  PARM03      : MQ REPLYTOQ                                         *
//*  PARM04      : MQ REPLYTOQMGR                                      *
//*                                                                    *
//**  PARMS TYPE=3  ***************************************************
//*                                                                    *
//*  PARM01      : FEPI POOL NAME                                      *
//*  PARM02      : FEPI TARGET NAME                                    *
//*  PARM03      : FEPI (BE) TIMEOUT VALUE                             *
//*  PARM04      : FEPI EXIT ACTION (R-RELEASE; F-FORCE; P-PASS;       *
//*                                  H-HOLD; A-LEAVE ASSIGNED)         *
//*  PARM05      : FEPI SIGNON PROGRAM NAME  (NOT CURRENTLY SUPPORTED) *
//*  PARM06      : FEPI SIGNOFF PROGRAM NAME (NOT CURRENTLY SUPPORTED) *
//*  PARM07      : FEPI ROUTING PROGRAM NAME (NOT CURRENTLY SUPPORTED) *
//*  PARM08      : FEPI NON-UNIQUE USERID                              *
//*                  (Y - NON-UNIQUE USERIDS IN USE.                   *
//*                       UNIQUE ASSIGNMENT REQUIRED.)                 *
//*                  (N - UNIQUE USERIDS IN USE                        *
//*                       NO UNIQUE ASSIGNMENT REQUIRED.)              *
//*                                                                    *
//**  PARMS TYPE=4  ***************************************************
//*                                                                    *
//*  PARM01      : MQ WAIT LIMIT (SECONDS)                             *
//*  PARM02      : MQ REPLYTOQ                                         *
//*  PARM03      : MQ REPLYTOQMGR                                      *
//*                                                                    *
//**  PARMS TYPE=5  ***************************************************
//*                                                                    *
//*  PARM01      : LINK3270 SERVICE NAME                               *
//*  PARM02      : LINK3270 FACILITYLIKE                               *
//*  PARM03      : LINK3270 MAX. FACILITY KEEPTIME (seconds)           *
//*  PARM04      : LINK3270 GETWAITINTERVAL (milli-seconds)            *
//*  PARM05      : LINK3270 DEALLOCATE FACILITY ON EXIT                *
//*                         (0-NO; 1-ALWAYS; 2-IF SUCCESSFUL;          *
//*                          3-IF UNSUCCESSFUL)                        *
//*  PARM06      : LINK3270 NON-UNIQUE USERID ASSIGNMENT               *
//*                         (Y - NON-UNIQUE USERIDS IN USE.            *
//*                              UNIQUE ASSIGNMENT REQUIRED.)          *
//*                         (N - UNIQUE USERIDS IN USE                 *
//*                              NO UNIQUE ASSIGNMENT REQUIRED.)       *
//*  PARM07      : LINK3270 AOR ROUTING ALLOWED/IN USE                 *
//*                         (Y - AOR ROUTING ALLOWED/IN USE)           *
//*                         (N - AOR ROUTING NOT ALLOWED/IN USE)       *
//*  PARM08      : LINK3270 VECTOR LOGGING  (0-OFF; 1-ON; 2-TRACE)     *
//*                                                                    *
//********************************************************************
//SYSIN   DD *
MODE=SAFE
TYPE=R
NAME=requestname
PARM01=requesttype
PARM02=initialprogramid
PARM03=initialtranid
PARM04=persistenceind
PARM05=xmlparseprgramid
PARM06=deploymentind
```

```
PARM07=initialprogramtype
PARMXX
TYPE=1
NAME=dplprogramid
PARM01=linkprogramid
PARM02=linksysid
PARM03=linktranid
PARM04=linksynconreturn
PARMXX
TYPE=2
NAME=mqputprogramid
PARM01=mqmdmsgtype
PARM02=mqodobjectname
PARM03=mqmdreplytoq
PARM04=mqmdreplytoqmgr
PARMXX
TYPE=3
NAME=fepiprogramid
PARM01=poolname
PARM02=targetname
PARM03=timeout
PARM04=releasetype
PARM05=RESERVED
PARM06=RESERVED
PARM07=RESERVED
PARM08=nonuniqueuserid
PARMXX
TYPE=4
NAME=mqgetprogramid
PARM01=mqgmowaitinterval
PARM02=mqodobjectname
PARM03=mqodobjectqmgrname
PARMXX
TYPE=5
NAME=link3270programid
PARM01=servicename
PARM02=facilitylike
PARM03=maxkeeptime
PARM04=getwaitinterval
PARM05=deallocatefacility
PARM06=nonuniqueuserid
PARM07=aorrouting
PARM08=vectorlogging
PARMXX
/*
//
```

Properties file dump JCL (DFHMAMPD)

```
//jobname  JOB DFHMAMPD,'DFHMAMPD',CLASS=A,MSGCLASS=X TYPRUN=SCAN
//*********************************************************************
//* @START_COPYRIGHT@
//*
//* 5655-M15
//*
//* CICS SERVICE FLOW RUNTIME
//*
//* @END_COPYRIGHT@
//*********************************************************************
//*********************************************************************
//* RUN DFHMADUP (CICS SFR PROPERTIES FILE DUMP PROGRAM)
//*
//*********************************************************************
//MADUP    EXEC PGM=DFHMADUP
//STEPLIB DD DSN='qual.SCIZLOAD',DISP=SHR
//*********************************************************************
//*     The output will go to SYSPRINT
```

```
//*******************************************************************
//SYSPRINT DD SYSOUT=X
//SYSOUT   DD SYSOUT=X
//DFHMAMPF DD DSN=qual.DFHMAMPF,DISP=SHR
//*
//
```

Properties file record descriptor (DFHMARPF)

```
*****************************************************************
*    CICS SFR PROPERTIES FILE RECORD DESCRIPTION
*****************************************************************
*****************************************************************
* @START_RRS_COPYRIGHT@
* Licensed Materials - Property of IBM
*
* "Restricted Materials of IBM"
*
* 5655-M15
*
* (c) Copyright IBM Corp. 2000, 2005
*
* CICS Service Flow Runtime
* @END_RRS_COPYRIGHT@
*****************************************************************
01  MIAC-PROPERTIES-RECORD.
    05  MIAC-PROPERTIES-KEY.
        10  MP-RECORD-IDENTIFIER              PIC X(08).
    05  MIAC-PROPERTIES-DATA.
        10  MP-STRUCID                        PIC X(04).
        10  MP-VERSION                        PIC S9(9)
        10  FILLER           PIC X(16)
        10  MP-DATA-FORMAT                    PIC S9(9) COMP.
        10  MP-DETAIL                         PIC X(220).
        10  MP-SYSTEM-AREA REDEFINES MP-DETAIL.
            15  MP-REQUESTTYPE                PIC S9(9) COMP.
            15  MP-INITIAL-PROGRAMID          PIC X(08).
            15  MP-INITIAL-TRANSID            PIC X(04).
            15  MP-PERSISTENCE-IND            PIC S9(9) COMP.
            15  MP-XML-PARSE-PROGRAMID        PIC X(08).
            15  MP-DEPLOYMENT-IND             PIC S9(9) COMP.
            15  MP-INITIAL-PROGRAMTYPE        PIC S9(9) COMP.
            15  FILLER PIC X(184).
        10  MP-DPL-AREA REDEFINES MP-DETAIL.
            15  MP-DPL-PROGRAM                PIC X(08).
            15  MP-DPL-SYSID                  PIC X(08).
            15  MP-DPL-MIRROR-TRANID          PIC X(04).
            15  MP-DPL-SYNCONRETURN           PIC X(01).
            15  FILLER                        PIC X(199).
        10  MP-FEPI-AREA REDEFINES MP-DETAIL.
            15  MP-FEPI-POOLNAME              PIC X(08).
            15  MP-FEPI-TARGETNAME            PIC X(08).
            15  MP-FEPI-TIMEOUT               PIC S9(9) COMP.
            15  MP-FEPI-TO-REDEFINES REDEFINES MP-FEPI-TIMEOUT.
                20  MP-FEPI-TO-OVERRIDE       PIC X(04).
            15  MP-FEPI-LOGOFFTYPE            PIC X(01).
            15  MP-FEPI-SIGNONPROGRAM         PIC X(08).
            15  MP-FEPI-SIGNOFFPROGRAM        PIC X(08).
            15  MP-FEPI-ROUTINGPROGRAM        PIC X(08).
            15  MP-FEPI-NONUNIQUE-USER        PIC X(01).
            15  FILLER                        PIC X(174).
        10  MP-MQPUT-AREA REDEFINES MP-DETAIL.
            15  MP-MQ-MSGTYPE                 PIC S9(9) COMP.
            15  MP-MQ-REQUESTQNAME            PIC X(48).
            15  MP-MQ-REPLYTOQ                PIC X(48).
            15  MP-MQ-REPLYTOQMGR             PIC X(48).
            15  FILLER                        PIC X(72).
```

```
10  MP-MQGET-AREA REDEFINES MP-DETAIL.
    15  MP-MQ-REPLYTOQNAME             PIC X(48).
    15  MP-MQ-REPLYTOQMGRNAME          PIC X(48).
    15  MP-MQ-WAITLIMIT                PIC S9(9) COMP.
    15  FILLER                         PIC X(120).
10  MP-BRIDGE-AREA REDEFINES MP-DETAIL.
    15  MP-BR-SERVICENAME              PIC X(16).
    15  MP-BR-FACILITYLIKE             PIC X(04).
    15  MP-BR-KEEPTIME                 PIC S9(8) COMP.
    15  MP-BR-GETWAITINTERVAL          PIC S9(8) COMP.
    15  MP-BR-DEALLOCATE-IND           PIC S9(4) COMP.
    15  MP-BR-NONUNIQUE-USERID         PIC X(01).
    15  MP-BR-AOR-ROUTING-IND          PIC X(01).
    15  MP-BR-VECTOR-LOGGING           PIC S9(4) COMP.
    15  FILLER                         PIC X(186).
```

Error file dump JCL (DFHMAMED)

```
//jobname  JOB DFHMAMED,'DFHMAMED',CLASS=A,MSGCLASS=X TYPRUN=SCAN
//*******************************************************************
//* @START_COPYRIGHT@
//*
//* 5655-M15
//*
//* CICS SERVICE FLOW RUNTIME
//*
//* @END_COPYRIGHT@
//*******************************************************************
//*******************************************************************
//* RUN DFHMAEUP (CICS SFR ERROR FILE DUMP PROGRAM)
//*
//*******************************************************************
//MAEUP    EXEC PGM=DFHMAEUP
//STEPLIB DD DSN='qual.SCIZLOAD',DISP=SHR
//*******************************************************************
//*     The output will go to SYSPRINT
//*******************************************************************
//SYSPRINT DD SYSOUT=X
//SYSOUT   DD SYSOUT=X
//DFHMAERF DD DSN=qual.DFHMAERF,DISP=SHR
//*
//
```

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 Repository file update JCL (DFHMAMLU)

The DFHMAMLU sample JCL utility is provided to support the running of MQIAC flows in the runtime environment only. It is used to generate repository entries from CICS BMS map source code. The sample JCL utility runs the Link3270 Repository file update batch program (DFHMALUP) that generates repository entries. This can be included as an additional step in a BMS assembly procedure. The Link3270 Repository file is not used in the runtime processing of CICS SFR service adapters.

```
//jobname  JOB DFHMAMLU,'DFHMAMLU',CLASS=A,MSGCLASS=H TYPRUN=SCAN
//*********************************************************************
//* @START_COPYRIGHT@
//*
//* 5655-M15
//*
//* CICS SERVICE FLOW RUNTIME
//*
//* @END_COPYRIGHT@
//*********************************************************************
//*********************************************************************
//* RUN DFHMALUP (CICS SFR REPOSITORY FILE UPDATE PROGRAM)
//*
//*********************************************************************
//MALUP    EXEC PGM=DFHMALUP
//STEPLIB DD DSN='hlq.SCIZLOAD',DISP=SHR
//*********************************************************************
//*     The output will go to SYSPRINT
//*********************************************************************
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//BMS      DD DSN=hlq.COBLIB(XXXXXXX),DISP=SHR
//DFHMALRF DD DSN=hlq.DFHMALRF,DISP=SHR
```

```
//*********************************************************************
//*                                                                   *
//*   SYSIN control card formats                                      *
//*                                                                   *
//**  PARAMETERS  ***************************************************
//*                                                                   *
//*  MAPSET=    Mapset name  (REQUIRED)                               *
//*             (7 bytes maximum; uppercase entry only)               *
//*                                                                   *
//*  OVERRIDE=  Y/N - use mapset name from MAPSET= parameter          *
//*                   overriding mapset name in BMS file.             *
//*             (1 byte OPTIONAL parameter; uppercase only if provided)*
//*                                                                   *
//*  STATIC=    Y/N - include fields with INITIAL data but no         *
//*                   field names. Static 3270 screen content.        *
//*             (1 byte OPTIONAL parameter; uppercase only if provided)*
//*                                                                   *
//*  PRINT=     Print switch; valid values (A/N/P/spaces).            *
//*                                                                   *
//*             (A)ll       - print all BMS records read and errors   *
//*             (N)o        - only errors reported                    *
//*             (P)rocessed - print only processed BMS records read   *
//*             spaces      - defaults to (P)                         *
//*             NULL        - defaults to (P)                         *
//*                                                                   *
//*             (1 byte OPTIONAL parameter; uppercase only if provided)*
//*                                                                   *
//*********************************************************************
//*+5++++0++++5++++0++++5++++0++++5++++0++++5++++0++++5++++0++++5++++0++
//SYSIN   DD *
MAPSET=name
//*OVERRIDE=
//*STATIC=
//*PRINT=
/*
//
```

Link3270 Vector Log file dump JCL (DFHMAMVD)

The following example of JCL shows the enhanced vector logging available with APAR PK32131 that uses two files, DFHMALVA and DFHMALVB, instead of the previous file DFHMALVF.

```
//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)
/*
//
```

Formatted file dumps

The following sections contain samples of formatted CICS Service Flow Runtime file dumps.

CICS Service Flow Runtime provides JCL to create formatted dumps of the following files:

- CICS Service Flow Runtime Properties file
- CICS Service Flow Runtime Error file
- CICS Service Flow Runtime Vector log

Properties file dump

Figure 47 on page 273 contains an example of the CICS Service Flow Runtime Properties file dump. See “Reading the Properties file dump” on page 273 for descriptions of the fields in the formatted Properties file dump.

For information on how to use the CICS Service Flow Runtime Properties file, see “Analyzing the properties file” on page 204.

For information on how to create the CICS Service Flow Runtime Properties file dump, see “Dumping the properties file” on page 99.

```
02/13/04                                CICS SFR Properties file (DFHMAMPF) Dump                                PAG
 ------------------------------------------------------------------------------------------------------------------------

  Property type:  R (Request properties)       Name:  DEMO_NAV
  Request type:  0 (Async)       Navigator name:  DEMOFNAV     Navigator transid:  DF01
  Property type:  3 (FEPI node)                Name:  DEMOBRW
  Pool name:  CICSDEV2     Target name:                Timeout value:          25  seconds   Exit action:  R (Release)
  Non-unique user:  N
  Property type:  3 (FEPI node)                Name:  DEMOINQ
  Pool name:  CICSDEV2     Target name:                Timeout value:          25  seconds   Exit action:  R (Release)
  Non-unique user:  N
  Property type:  3 (FEPI node)                Name:  DEMOSGOF
  Pool name:  CICSDEV2     Target name:                Timeout value:          25  seconds   Exit action:  Override
  Non-unique user:  N
  Property type:  3 (FEPI node)                Name:  DEMOSGON
  Pool name:  CICSDEV2     Target name:                Timeout value:          25  seconds   Exit action:  A (Leave assig
  Non-unique user:  N
  Property type:  1 (DPL node)                 Name:  DFHMAIP2
  Linked-to program:  DFHMABP4     Remote system name:  MAD2     Mirror transaction:           SYNCONRETURN:  N
  Property type:  2 (MQSeries node (PUT))      Name:  DFHMAIP3
  MQMD MsgType:  1 (Request)      Request queue:  CIA.IVP.DFHMABP6.REQUEST.QUEUE
  MQMD ReplyToQ:  CIA.IVP.DFHMABP6.REPLY.QUEUE                     MQMD ReplyToQMgr:
  Property type:  4 (MQSeries node (GET))      Name:  DFHMAIP4
  MQGMO WaitInterval:          30  seconds
  MQMD ReplyToQ:  CIA.IVP.DFHMABP6.REPLY.QUEUE                     MQMD ReplyToQMgr:
  Property type:  3 (FEPI node)                Name:  DFHMAIP5
  Pool name:  MQIACD2      Target name:                Timeout value:          20  seconds      Exit action:  R (Release)
  Non-unique user:  N
  Property type:  5 (Link3270 bridge node)     Name:  DFHMAIP6
  Service name:  (None)               Facilitylike:           Keeptime:   3600  secs     Getwaitinterval:     1  msecs
  Deallocate on exit:  1  (Always)              Non-unique userid:  N     AOR routing:  N     Vector logging:  ON
  Property type:  R (Request properties)       Name:  L3270REQ
  Request type:  1 (Sync)        Navigator name:  TESTL327     Navigator transid:  L327
  Property type:  R (Request properties)       Name:  MAIVPREQ
  Request type:  1 (Sync)        Navigator name:  DFHMAIP1     Navigator transid:  CMA5
  Property type:  5 (Link3270 bridge node)     Name:  PROGRAM1
  Service name:  (None)               Facilitylike:           Keeptime:   3600  secs     Getwaitinterval:  4200000  msecs
  Deallocate on exit:  1  (Always)              Non-unique userid:  N     AOR routing:  N     Vector logging:  ON
  Property type:  5 (Link3270 bridge node)     Name:  PROGRAM2
  Service name:  (None)               Facilitylike:           Keeptime:   3600  secs     Getwaitinterval:  4200000  msecs
  Deallocate on exit:  1  (Always)              Non-unique userid:  N     AOR routing:  N     Vector logging:  ON
```

Figure 47. Dumped Properties file

Reading the Properties file dump

The following information describes the fields on the formatted CICS Service Flow Runtime Properties file dump.

In describing the contents of the formatted properties file dump, those fields that appear in **bold** are the fields on the formatted properties file dump. The fields that appear in (**PARENTHESES**) are the corresponding Properties file fields from the copybook DFHMARPF.

Property type: (MP-DATA-FORMAT):
Data

The record type indicating the properties associated with this record.

- R (Request properties) (MP-DATA-FORMAT = -1). See "Request properties" on page 274 for field descriptions.
- **1 (DPL node)**. See "DPL node" on page 277 for field descriptions.

- **2 (WebSphere MQ node (PUT))**. See “WebSphere MQ node (PUT) properties” on page 277 for field descriptions.
- **3 (FEPI node)**. See “FEPI node properties” on page 278 for field descriptions.
- **4 (WebSphere MQ node (GET))**. See “WebSphere MQ node (GET) properties” on page 280 for field descriptions.
- **5 (Link3270 node)**. See “Link3270 node properties” on page 280 for field descriptions.
- **UNKNOWN (MP-DATA-FORMAT = unknown value)**

UNKNOWN could indicate that the data in the field is not allowed.

Name : (MP-RECORD-IDENTIFIER):
Data

This is the key for the property type record

If Property type indicates a request (R) record, this is a valid request name that can be processed by the CICS Service Flow Runtime at run time. A request record is read in the CICS Service Flow Runtime DPL Stub program (DFHMADPL) based upon the value in the DFHMAH structure of the request message. For more information on the for more information the **DFHMAH-REQUESTNAME** field, see “DFHMAH header structure” on page 80.

If **Property type** indicates a server adapter record (1, 2, 3, 4), this should be the name of a deployed and compiled server adapter. The record is read in the server adapter to retrieve the program properties used in it's execution. MQPUT and GET server adapter name values differ in the last character. An MQPUT server adapter name will have as it's last character a **P**. The associated MQGET server adapter name will have as it's last character a **G**

Request properties: The following are request properties:

Request type : (MP-REQUESTTYPE):
Data

Indicates the request processing mode for this request.

Valid values are:

- 0 (Async) Asynchronous
- 1 (Sync) Synchronous
- 2 (Sync RB) Synchronous rollback
- Unknown value

Asynchronous processing mode
indicates that at execution time the CICS Service Flow Runtime BTS process and all activities within the process are initiated asynchronously from it's parent (i.e., **BTS RUN ACTIVITY( ) ASYNCHRONOUS** command). This results in multiple units-of-work (UOW) each with a distinct commit scope.

Synchronous processing mode
indicates that at execution time the CICS Service Flow Runtime BTS process and all activities within the process are initiated synchronously with it's parent (i.e., **BTS RUN ACTIVITY( ) SYNCHRONOUS** command). This results in a context switch when the process and activities are initiated but the entire process is run as a single unit-of-work .

Synchronous rollback processing
is an implementation of the synchronous processing mode. However, when an error or failure occurs in adapter request processing, the unit-of-work is not committed. The CICS Service Flow Runtime issues an **EXEC CICS ABEND** command with the abend code equal to CIAX.

See CICS Business Transaction Services manual for more information on asynchronous and synchronous activations. See "Synchronous and asynchronous processing" on page 127 for an explanation of how these processing modes work at run time.

Nav/Init name: (MP-INITIAL-PROGRAMID):
Data

Indicates either the program name that should be executed for this request. This program name will be of the adapter Navigator (if the **Deployment** field indicates COMPLEX) or it will be of the server adapter (if the **Deployment** field indicates SIMPLE). The value is used on the PROGRAM parameter of the BTS DEFINE ACTIVITY command in the CICS Service Flow Runtime Navigation Manager (DFHMAMGR).

For further information on complex and simple deployment pattern processing, see "Deployment patterns" on page 53.

Note: WebSphere MQ server adapters are not supported when Deployment indicates (SIMPLE).

Nav/Init transid: (MP-INITIAL-TRANSID):
Data

Indicates the transaction Id of the adapter Navigator defined in the Nav/Init name field if the **Deployment** field indicates (COMPLEX) or it can indicate the transaction Id of the server adapter if the **Deployment** field indicates (SIMPLE). The value is used on the TRANSID parameter of the BTS DEFINE ACTIVITY command in the CICS Service Flow Runtime Navigation Manager (DFHMAMGR).

For further information on complex and simple deployment pattern processing, see "Deployment patterns" on page 53.

Type: (MP-INITIAL-PROGRAMTYPE):
Data

Indicates the deployed the program type that should be executed for this request.

The deployed program name is an adapter Navigator if the **Deployment** field indicates (COMPLEX)

The deployed program name is a server adapter if the **Deployment** field indicates (SIMPLE)

This value is used internally in the CICS Service Flow Runtime Navigation Manager (DFHMAMGR) to determine the correct CICS Service Flow Runtime BTS data container names to read/write during run time server processing. Valid display values are:

- 0 (Navigator)
- 1 (DPL)

- 3 (FEPI)
- 5 (Link3270)

For further information on complex and simple deployment pattern processing, see “Deployment patterns” on page 53.

Note: WebSphere MQ server adapters are not supported when Deployment indicates (SIMPLE).

Persistence: (MP-PERSISTENCE-IND):
Data

Indicates if the NOCHECK option is used on the BTS DEFINE PROCESS command in the DPL stub program DFHMADPL. Values are as follows:

- 0 (No) - indicates the NOCHECK option is used.
- 1 (Yes) - indicates the NOCHECK option is omitted.

For further information on complex and simple deployment pattern processing, see “Deployment patterns” on page 53.

For information on the NOCHECK option, see the *CICS Business Transaction Services* manual.

XML parse name: (MP-XML-PARSE-PROGRAMID):
Data

Indicates XML parse program name that is called by the CICS Service Flow Runtime DPL stub program (DFHMADPL) to parse the XML formatted application request data. If not specified, the application request data is not parsed and the XML formatted data is passed, in the input data container, to the deployed adapter flow.

Deployment: (MP-DEPLOYMENT-IND):
Data

Indicates the deployment pattern type of the deployed adapter flow, either SIMPLE (single connector type) or COMPLEX (aggregate connector type).

Adapter services of the aggregate connector type always execute with an adapter Navigator as part of the deployed flow.

Simple connectors execute 1 and only 1 of a FEPI, DPL or Link3270 server adapter program type. This value is used internally in the runtime environment.

Valid display values are as follows:

- 1 (SIMPLE)
- 2 (COMPLEX)

For further information on complex and simple deployment pattern processing, see “Deployment patterns” on page 53.

Note: WebSphere MQ server adapters are not supported when Deployment indicates (SIMPLE).

DPL node: See the *CICS Application Programming Reference* manual for more information on the EXEC CICS LINK command and the Distributed Program Link (DPL).

Linked-to program : (MP-DPL-PROGRAM):
Data

Indicates the DPL target and server program name. This is the PROGRAM parameter value specified on the EXEC CICS LINK command.

Remote system name : (MP-DPL-SYSID):
Data

Specifies the system name of the CICS server region to where the program link request is to be routed. This is the SYSID parameter value specified on the EXEC CICS LINK command.

Mirror transaction : (MP-DPL-MIRROR-TRANID):
Data

Indicates, if provided, the name of the mirror transaction that the remote region is to attach, and under which it is to run the server program. This is the TRANSID parameter value specified on the EXEC CICS LINK command.

SYNCONRETURN : (MP-DPL-SYNCONRETURN):
Data

Indicates if the server region named on the **SYSID** option is to take a syncpoint on successful completion of the server program.

Valid values are **Y** and **N**.

The value determines whether at execution time the SYNCONRETURN parameter is used on the EXEC CICS LINK command; Y = yes, N = no.

WebSphere MQ node (PUT) properties: See the *WebSphere MQ Application Programming Reference* for more information on WebSphere MQ and WebSphere MQ Integrator calls.

MQMD MsgType: (MP-MQ-MSGTYPE):
Data

Indicates the type of message being Put to the defined **Request queue**. The value is placed in the MQMD Message Descriptor structure field **MsgType** on the MQPUT1 call.

Valid values with the corresponding WebSphere MQ attribute are:

- 1 (Request) (MQMT_REQUEST)

  The message is one that requires a reply. The name of the queue and the name of the queue manager to which the reply should be sent must be specified in the ReplyToQ and the ReplyToQMgr fields respectively.
- 2 (Reply) (MQMT_REPLY)

  The message is the reply to an earlier request message (MQMT_REQUEST). The message should be sent to the queue indicated by the ReplyToQ field and the queue manager indicated by the ReplyToQMgr of the request message.

- 8 (Datagram) (MQMT_DATAGRAM)
  The message is one that does not require a reply.
- UNKNOWN (unknown value)

A value of 2 (REPLY) or 8 (Datagram), will not generate a Get program or the service flow code to invoke a Get program. A value of 2 (REPLY), indicates that the reply would be targeted to the application that issued the request.

See **MQMD Message Descriptor** in the *WebSphere MQ Application Programming Reference*.

Request queue : (MP-MQ-REQUESTQNAME):
Data

Indicates the queue name on which the message is *Put*. This value is placed in the MQOD Object Descriptor structure field **ObjectName** on the MQPUT1 call.

MQMD ReplyToQ: (MP-MQ-REPLYTOQ):
Data

Indicates the queue name on which the reply message is *Put* by the back-end WebSphere MQ-enabled application that performed the *Get* from the defined Request queue. This value is placed in the MQMD Message Descriptor field **ReplyToQ** on the MQPUT1 call.

MQMD ReplyToQMgr : (MP-MQ-REPLYTOQMGR):
Data

Indicates the queue manager name on which the reply message is *Put* by the back-end WebSphere MQ-enabled application that performed the *Get* from the defined Request queue. This value is placed in the MQMD Message Descriptor field **ReplyToQMgr** on the MQPUT1 call. MQMD ReplyToQ is the local name of a queue that is defined on this queue manager.

FEPI node properties: See the *CICS Front End Programming Interface User's Guide (FEPI)* for more information on FEPI, FEPI resources and FEPI APIs.

Pool name: (MP-FEPI-POOLNAME):
Data

The pool name used in the FEPI ALLOCATE POOL command to establish the connection and conversation with a target application. This is the POOL parameter value specified on the EXEC CICS FEPI ALLOCATE POOL command. If Pool name = **OVERRIDE**, the actual value is determined by the modeled adapter flow.

Non-unique user: (MP-FEPI-NONUNIQUEUSER):
Data

Indicates whether or not LU assignment processing for non-unique UserIDs was enabled for the FEPI navigator. See “FEPI adapter LU assignment processing for non-unique UserIDs” on page 158 for more information.

Target name : (MP-FEPI-TARGETNAME):

Data

This is the target name within the defined pool for which a conversation and connection should be established or attempted to be established. If specified, this is the TARGET parameter value specified on the EXEC CICS FEPI ALLOCATE POOL command. If Target name = **OVERRIDE**, the actual value is determined by the modeled adapter flow.

Timeout value: (MP-FEPI-TIMEOUT):
Data

This is a 3 byte field that indicates the approximate time, expressed in seconds, that the FEPI ALLOCATE POOL is to wait for a suitable session to become available, or the amount of time that a FEPI RECEIVE DATASTREAM or FEPI CONVERSE DATASTREAM is to wait for the requested data to arrive. If Timeout value = **Override**, the actual value is determined by the modeled adapter flow.

Exit action : (MP-FEPI-LOGOFFTYPE):
Data

This is a 1 byte field that indicates the action, if any, that should be taken on the connection or conversation when exiting the FEPI Navigator. This indicator also determines and controls processing related to the SLU Connection and Target Interaction files.

Valid values for the dump are:

- R (Release) (MP-FEPI-LOGOFFTYPE = 'R')
- P (Pass) (MP-FEPI-LOGOFFTYPE = 'P')
- F (Force) (MP-FEPI-LOGOFFTYPE = 'F')
- A (Leave assigned) (MP-FEPI-LOGOFFTYPE = 'A')
- H (Hold) (MP-FEPI-LOGOFFTYPE = 'H')
- Override (MP-FEPI-LOGOFFTYPE = 'O')

Release indicates that the connection should be set to released, ACQSTATUS = DFHVALUE (RELEASED) on FEPI SET CONNECTION command, and the conversation should be released, RELEASE option on the FEPI FREE command.

Force instructs FEPI to end the conversation unconditionally, FORCE option on the FEPI FREE command. An attempt is made to reset the connection the conversation was using.

Pass and ***Leave assigned*** indicate that the task is relinquishing ownership of the conversation in order for another task to acquire it, PASS option on the FEPI FREE command. There is no change in the processing state of the conversation. The difference in processing between *Pass* and *Leave assigned* is whether the connection is left assigned to the user when exiting the FEPI Navigator, see “CICS Service Flow Runtime FEPI file processing” on page 157 for information on how CICS Service Flow Runtimeprocesses FEPI file data.

Hold instructs FEPI to end the conversation but retain the session for use by another conversations. HOLD option on the FEPI FREE command. An attempt is made to reset the connection the conversation was using.

If Exit action = **Override**, the actual value is determined by the modeled adapter flow.

See the FEPI FREE command in the *CICS Front End Programming Interface User's Guide (FEPI)*.

WebSphere MQ node (GET) properties: See the *WebSphere MQ Application Programming Reference* for more information on WebSphere MQ and MQI calls.

MQGMO WaitInterval: (MP-MQ-WAITLIMIT) :
Data

This is a 3 byte field that indicates the approximate time, expressed in seconds, that the MQGET call waits for a suitable message to arrive. This value is used to calculate the number of milliseconds and is placed in the MQGMO Get Message options structure field **WaitInterval** on the MQGET call. A value of +999 indicates an unlimited wait interval (i.e., special value, MQWI_UNLIMITED). If the value can not be determined, UNKNOWN will appear on the dump.

MQMD ReplyToQ: (MP-MQ-REPLYTOQNAME):
Explanation

Indicates the queue name on which the reply message is *Put* by the back-end WebSphere MQ-enabled application that performed the *Get* from the defined Request queue. This value is placed in the MQOD Object Descriptor field **ObjectName** on the MQI MQGET call. In the current version of CICS Service Flow Runtime, this value should be the same as the corresponding field on the associated PUT server adapter properties record.

MQMD ReplyToQMgr: (MP-MQ-REPLYTOQMGRNAME):
Explanation

Indicates the queue manager name on which the reply message is Put by the back-end WebSphere MQ-enabled application that performed the Get from the defined Request queue. This value is placed in the MQOD Object Descriptor field **ObjectQMgrName** on the MQI MQGET call. MQMD ReplyToQ is the local name of a queue that is defined on this queue manager. In the current version of CICS Service Flow Runtime, this value should be the same as the corresponding field on the associated PUT server adapter properties record.

MQMD ReplyToQMgr : (MP-MQ-REPLYTOQMGR):
Explanation

Indicates the queue manager name on which the reply message is *Put* by the back-end WebSphere MQ-enabled application that performed the *Get* from the defined Request queue. This value is placed in the MQMD Message Descriptor field **ReplyToQMgr** on the MQPUT1 call. MQMD ReplyToQ is the local name of a queue that is defined on this queue manager.

Link3270 node properties: See the *CICS External Interfaces Guide Version 2 Release 2* and *Version 2 Release 3* (or higher) for more information on the Link3270 bridge mechanism, Link3270 message header (BRIH) and inbound and outbound Link3270 vectors.

Note: *Non-unique user/userid* and *AOR routing indicator* are not used in CICS Service Flow Runtime V3.1 server run-time processing. They are provided for prior product version (MQSeries Integrator Agent for CICS Transaction Server version 1.13) support only.

Service name : (MP-BR-SERVICENAME):

Explanation

This is a 16 byte field that indicates the service name used in the Link3270 server adapter for Link3270 State file processing. This is an optional user-defined parameter. It can be used to provide part of the Link3270 State file key if it is desirable to have multiple bridge facilities allocated, remain allocated and in use for a specific userid in adapter server processing. Facility session state information can be retained for a facility for a specific userid for use in the processing of one adapter server request (a service) while other facilities for that same specific userid may be in use in the processing of other adapter server requests (other services) concurrently, for example, multiple facilities allocated and in use for the same userid concurrently where session state information must be retained.

BRIH Facilitylike : (MP-BR-FACILITYLIKE) :
Explanation

This is a 4 byte field that indicates the name of an installed terminal that is to be used as a model for the bridge facility when allocated in a Link3270 server adapter. This is an optional parameter. If no value is specified, a CICS-supplied definition, **CBRF**, is used. The *facilitylike* value is the name of a real terminal resource definition that is used as a template for some of the properties of the bridge facility. Once the bridge facility is allocated this parameter is ignored by CICS on subsequent calls to the Link3270 bridge mechanism from your Link3270 server adapter.

See the *CICS External Interfaces Guide Version 2 Release 2* for further information.

BRIH Facilitykeeptime : (MP-BR-KEEPTIME):
Explanation

Bridge facilities are deleted after they have been unused for a given length of time. This is an optional parameter. This timeout value can be specified for use in a request to create a bridge facility for use in Link3270 server adapter(s). This parameter specifies the maximum timeout value in seconds. If this value is larger than the BRMAXKEEPTIME SIT parameter value in the AOR (router region), then CICS will change this parameter in the link parameter list (but does not return the reduced value).

Use this parameter to ensure that CICS Service Flow Runtime and CICS do not run short of bridge facilities caused by adapter flows not de-allocating bridge facilities due to adapter processing errors. Setting this parameter ensures bridge facilities are not reserved for too long a period. Alternatively, do not specify this parameter too small causing bridge facilities to be deleted when not expected or desired. This could cause your CICS Service Flow Runtime flows to function incorrectly.

The default is BRIHKT-DEFAULT (value = 86400 seconds). See the *CICS External Interfaces Guide Version 2 Release 2* for further information.

BRIH Getwaitinterval: (MP-BR-GETWAITINTERVAL) :
Explanation

This field indicates, for the back-end application transaction, the maximum wait interval for message input (in milliseconds) from the indicated Link3270 server adapter program. The maximum wait interval value used is the smaller of the BRIH-GETWAITINTERVAL and the RTIMEOUT value of the back-end application

transaction. Do not specify this parameter too small. This could cause your CICS Service Flow Runtime adapter flows to function incorrectly.

The default is BRIHGWI-MAXWAIT (value = 4200000 milliseconds). See the *CICS External Interfaces Guide Version 2 Release 2* for further information.

Facility delete indicator: (MP-BR-DEALLOCATE-IND):
Explanation

This is 1 byte field that indicates the action, if any, that should be taken on the allocated facility when exiting the Link3270 server adapter. This indicator also determines and controls processing related to the CICS Service Flow Runtime Link3270 State file.

Valid values are:

- 0 - Do not deallocate the facility when exiting (state data saved)
- 1 - Always deallocate the facility when exiting (no session state data saved)
- 2 - Deallocate the facility only if Link3270 server adapter processing was successful (session state data saved if processing was unsuccessful)
- 3 - Deallocate the facility only if Link3270 server adapter processing was unsuccessful (session state data saved if processing was successful)

Non-unique user/userid: (MP-BR-NONUNIQUE-USERID):
Explanation

This is a 1 byte field that indicates whether or not non-unique UserIDs are in use. It is not used in CICS Service Flow Runtime. This property is supported for compatibility with previous releases. See “Link3270 bridge facility assignment processing for non-unique user ids” on page 366 for more information on non-unique UserIDs.

AOR routing indicator : (MP-BR-AOR-ROUTING-IND) :
Data

This is a one byte field that indicates whether or not transactions that will run using the Link3270 bridge mechanism in the Link3270 server adapter can be routed, either dynamically or as defined on the transaction install definition. It is not used in CICS Service Flow Runtime V3.1. This property is supported for compatibility with previous releases only. Valid values are:

- Y - Yes, transactions can be routed
- N - No, transactions can not be routed

Note: If all the transactions in this Link3270 server adapter are defined to run in the same CICS region, not necessarily the AOR router region (region where this Link3270 Navigator will be executing), then this indicator should be set to 'N' to reduce performance overhead. If you do not know where the transactions will run, you should set this indicator to 'Y'. Although a value of 'Y' will increase performance overhead, it will ensure proper Link3270 server adapter execution. See information on **Transaction routing considerations** and **Link3270 bridge load balancing** in the *CICS External Interfaces Guide*. See “Configuring the runtime environment to use transaction routing” on page 165 for information on AOR transaction routing and configuring CICS Service Flow Runtime for transaction routing.

Bridge vector logging: (MP-BR-VECTOR-LOGGING):

Explanation

This is a 1 byte field that indicates whether Link3270 bridge inbound and outbound vectors should be logged in this Link3270 server adapter. The bridge vectors are logged in the Link3270 Vector log file (DFHMALVF). Vector logging is intended for use during Link3270 server adapter development for diagnostics and troubleshooting purposes.

Valid values are:

- 0 - Vector logging is off
- 1 - Vector logging is on

  This results in the logging of the Link3270 BRIH header structure, BRIV inbound and outbound vectors including any vector data. The vector data could be an inbound or outbound application data structure (ADS), text or 3270 datastream.
- 2 - Vector logging trace is on

  This results in the logging of the Link3270 BRIH header structure and BRIV inbound and outbound vectors minus any vector data. No vector ADS, text or 3270 datastream data is logged.

In passthrough processing, vector logging is determined by the value in field DFHMAH2-VECTOR-LOGGING in the passthrough header structure, DFHMAH2, in the passthrough request message. See “DFHMAH2 header structure” on page 87 for information on the DFHMAH2-VECTOR-LOGGING field.

Error file dump

Figure 48 on page 284 contains an example of the CICS Service Flow Runtime Error file dump. See “Reading the Error file dump” on page 285 for descriptions of the fields in the formatted dump.

For information on how to dump the error file, see the “Troubleshooting checklist” on page 202.

```
1   02/14/01                                  CICS SFR Error file (DFHMAERF) Dump                                  PAGE
 ------------------------------------------------------------------------------------------------------------------------

0 Processed:  Date:  02/08/01     Time:  08:39:51:       PutApplid:  CICSDEV1     PutTranid:  CMA5
     Error:  CIA05004     Normal processing

    Userid:  SYSMKW1      Applid:  CICSDEV1     Tranid:  MQ0G     Eibtaskn:   0000310     AbsTime:  003190610389060
   Request:  MAIVPREQ       Mode:  Sync        Program:  DFHMAIP4     Type:  MQGET
  Activity:  MAIVPIP4                       Node Name:  DFHMAIP4
     Event:  DFHINITIAL                    Event type:  System       Step:  GET
  Proctype:  DFHMAINA                        Process:  0031906103787201509
 Failed Processtype:                    Failed Process:
  ReplyToQ:                                                ReplyToQMgr:
  MQ MsgId:                                MQ CorrelId:

 Error detail:  MQSeries
 Completion code:   000000002     Reason code:   000002016     Object type:  000000001
 Object name:  CIA.IVP.DFHMABP6.REPLY.QUEUE                       QMgr name:
 Reslvd Qname:                                                    Reslvd QMgr:
 ------------------------------------------------------------------------------------------------------------------------
0 Processed:  Date:  02/14/01     Time:  08:27:06:       PutApplid:  CICSDEV1     PutTranid:  CKBP
     Error:  CIA01001     Normal processing

    Userid:  SYSMKW1      Applid:  CICSDEV1     Tranid:  CKBP     Eibtaskn:   0000660     AbsTime:  003191128022510
   Request:  MAIVPREQ       Mode:  UNKNOWN     Program:  DFHMADPL     Type:  System
  Activity:                                  Node Name:
     Event:                                 Event type:  None         Step:  PROPERTIES
  Proctype:  DFHMAINA                        Process:  0031911280207501503
 Failed Processtype:                    Failed Process:
  ReplyToQ:                                                ReplyToQMgr:
  MQ MsgId:                                MQ CorrelId:

 Error detail:  File
 File status:  00000019     File name:  DFHMAMPF     File function:  Start Browse
 File key data:  MAIVPREQ
 ------------------------------------------------------------------------------------------------------------------------

1   03/18/04                                  CICS SFR Error file (DFHMAERF) Dump                                  PAGE
 ------------------------------------------------------------------------------------------------------------------------

  Activity:  LB_NAV                          Node Name:  LB-RESET
     Event:  DFHINITIAL                     Event type:  System       Step:  INPUT
  Proctype:  DFHMAINA                        Process:  0032886283043330TC07
 Failed Processtype:                    Failed Process:
  ReplyToQ:  SIMULATION.BAEFB1B25F9B0689                   ReplyToQMgr:  MAQ1
  MQ MsgId:                                MQ CorrelId:

 Error detail:  Link3270
   ***  INBOUND RECEIVE MAP  (1802)  ***

            Structure:      BRHI
              Version:         0
     Structure length:       180
        Facilitytoken:                       (NEW)
              Netname:
             Terminal:
        Transactionid:  CMAW
           Datalength:       792
         Return code:         61  (Invalid facilitytoken    )
           Comp code:          1
         Reason code:          0
            Function:
           Abendcode:             (NONE)
        Taskendstatus:          0  (Conversation)
   Remainingdatalength:         0
     Nexttransactionid:             (NONE)
      Nexttranidsource:         0  (Normal)
           Erroroffset:         0
                 Segno:         0
    BRIV vector length:       612
    RM Xmit send areas:  NO
             RM Mapset:  DFHMAB1
```

Reading the Error file dump

The following information describes the fields on the formatted CICS Service Flow Runtime Error file dump.

The fields on the formatted Error file dump are derived from fields in the CICS Service Flow Runtime Error file (DFHMAERF).

The formatted Error file dump is organized into two portions as follows:

- A *static* portion
- A *dynamic* portion

In describing the contents of the formatted error file dump, those fields that appear in **bold** are the fields on the formatted Error dump. The fields that appear in (**PARENTHESES**) are the corresponding Error file fields from the copybook DFHMARER.

Static portion of the formatted error file dump

The fields that are displayed in the *static* portion of the error dump remain the same, regardless of the type of error reported. The static portion of the error dump contains information that ends at the **Error detail** field.

Processed: Date: (ERR-REC-DATE):
Data

The *actual* date that the record was written to the Error file. This date can be different than the date on which the error occurred if, instead of being immediately processed by the Error Listener program, the records were left on the error queue. This could happen if an unexpected event prevented the Error Listener program from immediately processing the error.

Processed: Time: (ERR-REC-TIME):
Data

The *actual* time that the record was written to the Error file. This time can be different than the time on which the error occurred if, instead of being immediately processed by the Error Listener program, the records were left on the error queue. This could happen if an unexpected event prevented the Error Listener program from immediately processing the error.

PutApplid / PutTranid: (ERR-REC-PUTAPPLID) / (ERR-REC-PUTTRANID):
Data

The APPLID and TRANID retrieved from the origin context in the WebSphere MQ Message Descriptor structure (MQMD), after a successful MQGET in the Error Listener.

When an error record is PUT to the CICS Service Flow Runtime Error queue, the queue manager loads the MQMD-PUTAPPLNAME with the name of the application that performed the PUT, in this case, the 8–byte CICS applid, followed by the 4–character tranid under which the application was running. The field MQMD-PUTAPPLNAME is moved to group item ERR-REC-PUTAPPLNAME in the Error Listener.

See the *WebSphere MQ Application Programming Reference* for more information.

Error: (ERR-REC-MESSAGE):
Data

The error message that indicates the *type* of error. See “Error messages” on page 220 for a description of the different types of error messages that can display in this field.

Error Description: (ERR-REC-UOWCONTROL):
Data

The mode (Normal, Compensation, Passthrough and CANCEL) of adapter request processing. The value is derived from a field in the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on the DFHMAH-UOWCONTROL field.

Valid values for the dump are:

- Normal processing (ERR-REC-UOWCONTROL = zero)
- Compensation processing (ERR-REC-UOWCONTROL = +2)
- Passthrough processing (ERR-REC-UOWCONTROL = +3)
- CANCEL processing (ERR-REC-UOWCONTROL = +9)

Userid: (ERR-REC-USERID):
Data

The user's id under which the error was reported, and for which the adapter request processing was occurring.

Applid / Tranid: (ERR-REC-APPLID) / (ERR-REC-TRANSACTION):
Data

The APPLID and TRANID indicating the CICS region and transaction, under which the failed application program was running, where the error actually occurred.

This can be different from the PutApplid if an error occurred in a DPL or WebSphere MQ server adapter that was not running in the same region as the controlling adapter Navigator (for example, due to Workload balancing). The DPL and WebSphere MQ server adapters do not do PUTs to the CICS Service Flow Runtime Error queue. Instead, they return control to the adapter Navigator indicating the error. The Navigator subsequently performs the PUT to the error queue.

Eibtaskn: (ERR-REC-EIBTASKN):
Data

The CICS EIBTASKN. The task number of the failed transaction assigned by CICS . See the *CICS Application Programming Reference* manual for more information.

AbsTime: (ERR-REC-ABSTIME):
Data

The time as reported by CICS when the error occurred. This time is retrieved via an EXEC CICS ASKTIME command. See the *CICS Application Programming Reference* manual for more information

Request: (ERR-REC-REQUESTNAME):

Data

The name of the request that was being processed when the failure occurred. This field corresponds to a TYPE = R record in the CICS Service Flow Runtime Properties file. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on the **DFHMAH-REQUESTNAME** field.

Mode: (ERR-REC-REQUESTTYPE):
Data

The type or mode (i.e., asynchronous, synchronous, synchronous rollback) of adapter request processing that was in effect when the failure occurred The request type value is found on a TYPE = R record in the CICS Service Flow Runtime Properties file and is retrieved based on the request name.

Valid values for the dump are:

- Async (ERR-REC-REQUESTTYPE = zero)
- Sync (ERR-REC-REQUESTTYPE = +1)
- Sync RB (ERR-REC-REQUESTTYPE = +2)

Program: (ERR-REC-PROGRAM-ID):
Data

The program name that reported the error or where the failure occurred. In the case of a DPL server adapter, it could be the DPL target program name. This would be the case if the error was an abend in the DPL target program.

Type: (ERR-REC-PROGRAM-TYPE):
Data

The type of program indicated by the program name. Valid values for the dump are:

- Navigator (ERR-REC-PROGRAM-TYPE = zero)
- DPL (ERR-REC-PROGRAM-TYPE = +1)
- MQPUT (ERR-REC-PROGRAM-TYPE = +2)
- FEPI (ERR-REC-PROGRAM-TYPE = +3)
- MQGET (ERR-REC-PROGRAM-TYPE = +4)
- LINK3270 (ERR-REC-PROGRAM-TYPE = +5)
- System (ERR-REC-REQUESTTYPE < zero)

A value of **system** would indicate that an CICS Service Flow Runtime module reported the error. For example, the CICS Service Flow Runtime DPL Stub program (DFHMADPL) or the CICS Service Flow Runtime Navigation Manager (DFHMAMGR).

Activity: (ERR-REC-ACTIVITY):
Data

The activity name that reported the error or where the failure occurred. In the current version of CICS Service Flow Runtime , the activity name is the program name being run as an activation. If the error was reported in the CICS Service Flow Runtime DPL Stub program (DFHMADPL), this field should be blank. For information on activity names, see the *CICS Business Transaction Services* manual.

Node Name: (ERR-REC-NODE-NAME):
Data

The node name that reported the error or where the failure occurred. In the current version of CICS Service Flow Runtime, the node name is the program name being run as an activation. Node names apply to generated and deployed server adapters. This field should be blank if the error occurred or is reported by an CICS Service Flow Runtime system module.

The node name is not used by CICS Service Flow Runtime at run time. For information on how nodes are used, see the applicable topics in the Service Flow Modeler help in the WebSphere Developer for System z information center.

Event: (ERR-REC-EVENT):
Data

The last reattach event retrieved at the time of failure. For information regarding BTS events, see the *CICS Business Transaction Services* manual

Event type: (ERR-REC-EVENTTYPE):
Data

The event type of the last reattach event *retrieved* at the time of failure. For information regarding BTS events and event types, see the *CICS Business Transaction Services* manual.

Valid values for the dump are:

- Input (ERR-REC-EVENTTYPE = +226)
- Activity (ERR-REC-EVENTTYPE = +1002)
- Composite (ERR-REC-EVENTTYPE = +1003)
- Timer (ERR-REC-EVENTTYPE = +1004)
- System (ERR-REC-EVENTTYPE = +643)
- None (ERR-REC-EVENTTYPE = zero)
- UNKNOWN (ERR-REC-EVENTTYPE = unknown value)

UNKNOWN could indicate that the data in the event type field is not allowed.

Step: (ERR-REC-STEP):
Data

Indicates the internal program step at the time of failure.

Valid values for the dump are:

- Initial
- Properties
- Define
- Main
- Run
- Check
- Output
- Reply
- Input

- Return
- Terminate
- Put1
- Child activity name
- Compensate
- Cancel

Proctype: (ERR-REC-PROCESSTYPE):
Data

The process type of the request or process that reported the error or failure. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure. See “DFHMAH field definitions” on page 81 for information on the DFHMAH-PROCESSTYPE field.

For information regarding BTS processes, see the *CICS Business Transaction Services* manual.

Process: (ERR-REC-PROCESSNAME):
Data

The process name of the CICS Service Flow Runtime request processing instance that reported the error or failure. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure. See “DFHMAH field definitions” on page 81 for information on the DFHMAH-PROCESSNAME.

For more information regarding BTS processes, see the *CICS Business Transaction Services* manual

Failed Processtype: (ERR-REC-FAIL-PROCTYPE):
Data

The failed process type of a previously run instance of the CICS Service Flow Runtime request processing that resulted in error or failure. This field will be reported when an error or failure occurs in a compensating flow process that was run as the result of an earlier adapter request processing failure. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on the DFHMAH-FAILED-PROCTYPE field.

For information regarding BTS processes, see the *CICS Business Transaction Services* manual.

Failed Process: (ERR-REC-FAIL-PROCNAME):
Data

The failed process name of a previously run instance of the CICS Service Flow Runtime request processing that resulted in error or failure. This field will be reported when an error or failure occurs in a compensating flow process that was run as the result of an earlier adapter request processing failure. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on the DFHMAH-FAILED-PROCNAME field. For information regarding BTS processes, see the *CICS Business Transaction Services* manual.

ReplyToQ : (ERR-REC-REPLYTOQ):
Data

The WebSphere MQ reply queue name where CICS Service Flow Runtime writes the error reply. This field will be reported when an error or failure occurs in asynchronous request processing. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on this field.

For information regarding WebSphere MQ programming, see *WebSphere MQ Application Programming Reference* and the *WebSphere MQ Application Programming Guide*.

ReplyToQMgr: (ERR-REC-REPLYTOQMGR):
Data

The WebSphere MQ reply queue manager name where CICS Service Flow Runtime writes the error reply. This field will be reported when an error or failure occurs in Asynchronous request processing. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on this field. For information regarding WebSphere MQ programming, see *WebSphere MQ Application Programming Reference* and the *WebSphere MQ Application Programming Guide*.

MQ MsgId: (ERR-REC-MSGID):
Data

The WebSphere MQ message identifier set by the service requestor for use on all MQI Puts in CICS Service Flow Runtime MQ server adapters and CICS Service Flow Runtime reply messages. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on this field. This field will be reported when an error or failure occurs in Asynchronous request processing and DFHMAH-MSGID is loaded by the service requestor. For information regarding WebSphere MQ programming, see *WebSphere MQ Application Programming Reference* and the *WebSphere MQ Application Programming Guide*.

MQ CorrelId: (ERR-REC-CORRELID):
Data

The WebSphere MQ correlation identifier set by the service requestor. It is derived from the CICS Service Flow Runtime message header (DFHMAH) structure in the request message. See “DFHMAH field definitions” on page 81 for information on this field. This field will be reported when an error or failure occurs in Asynchronous request processing and DFHMAH-CORRELID is loaded by the service requestor. For information regarding WebSphere MQ programming, see *WebSphere MQ Application Programming Reference* and the *WebSphere MQ Application Programming Guide*.

Dynamic portion of the formatted error file dump

The **Error detail** field marks the beginning of the dynamic portion of the formatted error dump. This means that the fields in this portion of the error dump will vary depending on the *type* of error reported.

The descriptions of fields in the dynamic portion of the formatted error file dump are documented in sections according to the error type.

Error types are categorized as follows:

- **File (ERR-REC-FORMAT = +1; ERR-REC-MESSAGE = CIA010nnE).** See "Dynamic portion of error dump — File errors" on page 292 for field descriptions.
- **Data Container (ERR-REC-FORMAT = +2; ERR-REC-MESSAGE = CIA020nnE)**. See "Dynamic portion of error dump — Data-container errors" on page 293 for field descriptions.
- **DPL (ERR-REC-FORMAT = +3; ERR-REC-MESSAGE = CIA030nnE)**. See "Dynamic portion of error dump — DPL errors" on page 294 for field descriptions.
- **FEPI (ERR-REC-FORMAT = +4; ERR-REC-MESSAGE = CIA040nnE)**. See "Dynamic portion of error dump — FEPI errors" on page 296 for field descriptions.
- **MQSeries (ERR-REC-FORMAT = +5; ERR-REC-MESSAGE = CIA050nnE)**. See "Dynamic portion of error dump — MQ errors" on page 298 for field descriptions.
- **BTS (ERR-REC-FORMAT = +6; ERR-REC-MESSAGE = CIA060nnE)**. See "Dynamic portion of error dump — BTS errors" on page 306 for field descriptions.
- **Link3270 (ERR-REC-FORMAT = +7; ERR-REC-MESSAGE = CIA070nnE)**. See "Dynamic portion of error dump — Link3270 errors" on page 298 for field descriptions.
- **TSQ (ERR-REC-FORMAT = +8; ERR-REC-MESSAGE = CIA013nnE)**. See "Dynamic portion of error dump — Temporary storage queue errors" on page 300 for field descriptions.
- **CICS (ERR-REC-FORMAT = +9; ERR-REC-MESSAGE = CIA081nnE)**. See "Dynamic portion of error dump — CICS API command errors" on page 301 for field descriptions.

  Note: Error messages **CIA07015E** and **CIA07016E** may also be classified as CICS API command errors if reported by the passthrough XML parse and build program, DFHMAXPI, when no additional Link3270 bridge header and vector information is available.
- **XML (ERR-REC-FORMAT = +10; ERR-REC-MESSAGE = CIA083nnE)**. See "Dynamic portion of error dump — XML parse or build errors" on page 302 for field descriptions.
- **Field (ERR-REC-FORMAT = +11; ERR-REC-MESSAGE = CIA08005E)**. See "Dynamic portion of error dump — Field validation error" on page 303 for a field description.

  Note: The field validation error is an application error where additional error detail is provided.
- **BIDI Transformation (ERR-REC-FORMAT = +12; ERR-REC-MESSAGE = CIA08008E)**. See "Dynamic portion of error dump - BIDI transformation errors" on page 310 for field descriptions
- **Application (ERR-REC-FORMAT = zero; ERR-REC-MESSAGE = CIA080nE)**. See "Dynamic portion of error dump — Application errors" on page 312 for field descriptions.
- **Abend (ERR-REC-FORMAT = +999; ERR-REC-MESSAGE = CIA99999E)**. See "Dynamic portion of error dump — Abend errors" on page 309 for field descriptions.
- **UNKNOWN (ERR-REC-FORMAT not = to one of the above values or ERR-REC-MESSAGE not identified)**

See "Error messages" on page 220 for a listing of the error messages.

Error detail: (ERR-REC-FORMAT):

Dynamic portion of error dump — File errors: The following field descriptions apply to the dynamic portion of the error file dump, when the error is a file error

File resp: (ERR-REC-FILE-RESP):
Data

Indicates the CICS RESP code returned for the **File function** performed. See the *CICS Application Programming Reference* for more information on valid CICS RESP codes and values used on indicated EXEC CICS APIs.

resp2: (ERR-REC-FILE-RESP2):
Data

Indicates the CICS RESP2 code returned for the **File function** performed. See the *CICS Application Programming Reference* for more information on valid CICS RESP2 codes and values used on indicated EXEC CICS APIs.

File name: (ERR-REC-FILE-NAME):
Data

Indicates the CICS Service Flow Runtime file name, (CICS DDNAME) for which the error occurred.

Valid values are:

- DFHMAMPF (CICS Service Flow Runtime Properties file; VSAM KSDS)
- DFHMATIF (CICS Service Flow Runtime FEPI Target Interaction file; VSAM KSDS)
- DFHMACOF (CICS Service Flow Runtime FEPI SLU Connection file; VSAM KSDS)
- DFHMAC1F (CICS Service Flow Runtime FEPI SLU Connection alternate file; VSAM KSDS)
- DFHMAL2F (CICS Service Flow Runtime Link3270 Business State file; VSAM KSDS)
- DFHMALVF (CICS Service Flow Runtime Link3270 Vector Log file; VSAM KSDS)

See Table 12 on page 111 for a description of the files.

File function: (ERR-REC-FILE-FUNCTION):
Data

Indicates the function attempted for the defined file that resulted in the error or I/O failure

Valid values are:

- Start Browse (CICS STARTBR with GTEQ option)
- Read Next (CICS READNEXT)
- Read (CICS READ)
- Read Update (CICS READ for UPDATE)
- Write (CICS WRITE)
- Rewrite (CICS REWRITE)
- Delete (CICS DELETE)

- Start Browse EQ (CICS STARTBR with GENERIC and EQUAL options)
- End Browse (CICS ENDBR)
- Unlock (CICS UNLOCK)
- Write RBA (CICS WRITE with RBA option; applies to VSAM ESDS data sets only (CICS Service Flow Runtime Error file (DFHMAERF))
- Direct Delete (CICS DELETE with RIDFLD and KEYLENGTH options)

See the *CICS Application Programming Reference* for more information.

Key length: (ERR-REC-FILE-KEY-LEN):
Data

Indicates the file key length value (KEYLENGTH option) for VSAM KSDS file used in the function attempted for the defined file that resulted in the error or I/O failure.

File length: (ERR-REC-FILE-LEN) :
Data

Indicates the record length value (LENGTH option) for VSAM KSDS file used in the function attempted for the defined file that resulted in the error or I/O failure.

File key data: (ERR-REC-FILE-KEY):
Data

Indicates the file key data value for VSAM KSDS file used in the function attempted for the defined file that resulted in the error or I/O failure.

Dynamic portion of error dump — Data-container errors: The following field descriptions apply to the dynamic portion of the error file dump, when the error is a data-container error.

See the *CICS Business Transaction Services* manual for more information on data-containers and the BTS container APIs GET and PUT.

See “CICS Service Flow Runtime data-containers for aggregate processing patterns” on page 115 for information on specific data-containers used by CICS Service Flow Runtime at run time.

CICS Resp : (ERR-REC-DC-RESP):
Data

Indicates the CICS RESP code returned on the data-container action performed. See the *CICS Application Programming Reference*l for more information on valid CICS RESP codes and values.

See the *CICS Business Transaction Services* manual for more information on data-containers and the BTS container APIs GET and PUT.

CICS Resp2: (ERR-REC-DC-RESP2):
Data

Indicates the CICS RESP2 code for CICS RESP returned on the data-container action performed. See the *CICS Business Transaction Services* manual for more information on valid CICS RESP2 codes and values on BTS data-container APIs.

See “CICS Service Flow Runtime data-containers for aggregate processing patterns” on page 115 for information on specific data-containers used by CICS Service Flow Runtime at run time.

Container name: (ERR-REC-DC-NAME):
Data

Indicates data-container name for which the error or failure occurred.

See “CICS Service Flow Runtime data-containers for aggregate processing patterns” on page 115 for information on specific data-containers used by CICS Service Flow Runtime at run time.

Owner: (ERR-REC-DC-OWNER):
Data

The name of the activity in which the data-container error occurred.

Container data: (ERR-REC-DC-DATA):
Data

The first 48 bytes of information in the data-container in question. If the action performed was a GET CONTAINER, it is the first 48 bytes retrieved. If the action performed was a PUT CONTAINER, it is the first 48 bytes of information to be placed in the data-container.

See “CICS Service Flow Runtime data-containers for aggregate processing patterns” on page 115 for information on specific data-containers used by CICS Service Flow Runtime at run time.

Dynamic portion of error dump — DPL errors: The following field descriptions apply to the dynamic portion of the error file dump, when the error is a DPL error.

See the *CICS Application Programming Reference* manual for more information on Distributed Program Links (DPL) and the EXEC CICS LINK command.

CICS Resp : (ERR-REC-DPL-RESP):
Data

Indicates the CICS RESP code returned on the EXEC CICS LINK command. See the *CICS Application Programming Reference* for more information on CICS RESP codes and values.

See the *CICS Application Programming Reference* for information on the Distributed Program Links (DPL) and the EXEC CICS LINK command.

CICS Resp2: (ERR-REC-DPL-RESP2):
Data

Indicates the CICS RESP2 code for CICS RESP returned on the EXEC CICS LINK command. See the *CICS Application Programming Reference* for more information on valid CICS RESP2 codes and values on the EXEC CICS LINK command

Linked-to program : (ERR-REC-DPL-PROGRAM):

Data

The DPL target linked-to program name. This is the PROGRAM parameter value specified on the EXEC CICS LINK command. This value is derived from the TYPE = 1 record, PARM01, in the CICS Service Flow Runtime Properties file. The specific TYPE = 1 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump.

Remote SYSID: (ERR-REC-DPL-SYSID):
Data

Specifies the system name of the CICS server region to where the program link request was routed. This is the SYSID parameter value specified on the EXEC CICS LINK command. This field can be blank if the LINK request is a normal local link request. This value is derived from the TYPE = 1 record, PARM02, in the CICS Service Flow Runtime Properties file. The specific TYPE = 1 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump.

Synconreturn: (ERR-REC-DPL-SYNCONRETURN):
Data

Indicates if the server region named on the SYSID option is to take a syncpoint on successful completion of the server program. Valid values are **Y** and **N**.

The value determines whether at execution time the SYNCONRETURN parameter is used on the EXEC CICS LINK command; Y - yes, N - no. This value is derived from the TYPE = 1 record, PARM04, in the CICS Service Flow Runtime Properties file. The specific TYPE = 1 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump.

Mirror transid : (ERR-REC-DPL-TRANSID):
Data

Indicates, if provided on the EXEC CICS LINK command, the name of the mirror transaction that the remote region is to attach, and under which it is to run the server program. If there is no TRANSID parameter specified on the EXEC CICS LINK command, the server region attaches default values of either CSMI, CPMI, or CVMI. This value is derived from the TYPE = 1 record, PARM03, in the CICS Service Flow Runtime Properties file. The specific TYPE = 1 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump.

COMMAREA length: (ERR-REC-DPL-LENGTH):
Data

Indicates the length in bytes of the COMMAREA (communication area). This is the LENGTH parameter value specified on the EXEC CICS LINK command.

Data length : (ERR-REC-DPL-DATALENGTH):
Data

Indicates the actual data length in bytes of the COMMAREA (communication area) passed to the linked-to program. This is the DATALENGTH parameter value specified on the EXEC CICS LINK command.

COMMAREA data : (ERR-REC-DPL-DATA):
Data

Specifies the first 48 bytes of the communication and data area that is passed to the invoked linked-to program. This is the COMMAREA parameter value specified on the EXEC CICS LINK command.

Dynamic portion of error dump — FEPI errors: The following field descriptions apply to the dynamic portion of the error file dump, when the error is a FEPI error.

See the *CICS Front End Programming Interface User's Guide (FEPI)* for more information on FEPI, FEPI resources and FEPI APIs.

CICS Resp : (ERR-REC-FEPI-RESP):
Data

Indicates the CICS RESP code returned on the EXEC CICS FEPI command. See the *CICS Application Programming Reference* for more information on valid CICS RESP codes and values.

See the *CICS Front End Programming Interface User's Guide (FEPI)* for more information on FEPI, FEPI resources and FEPI APIs.

CICS Resp2 : (ERR-REC-FEPI-RESP2):
Data

Indicates the CICS RESP2 code for CICS RESP returned on the EXEC CICS FEPI command. See the *CICS Front End Programming Interface User's Guide (FEPI)* for more information on valid CICS RESP2 codes and values.

Convid : (ERR-REC-FEPI-CONVID):
Data

The unique identifier of the FEPI conversation for which the error or failure occurred. This is the ID that is used on all subsequent commands for the conversation following the FEPI ALLOCATE POOL command. This field can be blank if the error or failure occurred before a conversation was established.

Transid : (ERR-REC-FEPI-TRANSID):
Data

The TRANID indicating the transaction of the FEPI server adapter where the error or failure occurred. In the current version of CICS Service Flow Runtime, this field indicates the same transaction indicated in the static Tranid field.

Pool name : (ERR-REC-FEPI-POOL-NAME):
Data

The pool name used in the FEPI ALLOCATE POOL command to establish the connection and conversation with a target application. This is the POOL parameter value specified on the EXEC CICS FEPI ALLOCATE POOL command. This value is derived from the TYPE = 3 record, PARM01, in the CICS Service Flow Runtime Properties file unless PARM01=OVERRIDE. The specific TYPE = 3 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump. If PARM01=OVERRIDE, the value is determined by the modeled adapter flow.

Target name: (ERR-REC-FEPI-TARGET-NAME):
Data

The target name within the defined pool for which a conversation and connection was established or attempted to be established. This field can be blank if the error or failure occurred before a connection or conversation was established. If specified, this is the TARGET parameter value specified on the EXEC CICS FEPI ALLOCATE POOL command. Also if specified, this value is derived from the TYPE = 3 record, PARM02, in the CICS Service Flow Runtime Properties file unless PARM02= OVERRIDE. The specific TYPE = 3 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump. If PARM02=OVERRIDE, the value is determined by the modeled adapter flow.

Target Applid : (ERR-REC-FEPI-TARGET-APPLID):
Data

The target applid with which a connection and conversation was established or attempted to be established. This field can be blank if the error or failure occurred before a conversation was established. This is the value specified on the target definition.

Node name : (ERR-REC-FEPI-NODE-NAME):
Data

The node name (simulated secondary LU terminal) within the defined pool for which a connection and conversation was established or attempted to be established with the defined target. This field can be blank if the error or failure occurred before a connection and conversation was established.

Node owner : (ERR-REC-FEPI-NODE-OWNER):
Data

The owner of the established connection and conversation. This field can be the signed-on user as established in the WebSphere MQ-CICS bridge or it can be provided by the modeled adapter flow. If established by the WebSphere MQ-CICS bridge, the value is passed in the MQMD Message Descriptor structure of the CICS Service Flow Runtime request message, see the *WebSphere MQ System Programming Reference* manual. This field can be blank if the error or failure occurred before the connection and conversation was established.

Propertyset : (ERR-REC-FEPI-PROPERTYSET):
Data

If there is a value, it indicates the installed PROPERTYSET name for the defined Pool name.

ESM Resp / ESM Reason: (ERR-REC-FEPI-ESM-RESP) / (ERR-REC-FEPI-ESM-REASON):
Data

Indicates the response and reason codes, respectively, returned by RACF or other External Security Manager on the EXEC CICS FEPI REQUEST PASSTICKET command. For RACF response and reason codes, see *OS/390 Security Server Messages and Codes* manual.

Dynamic portion of error dump — MQ errors: The following field descriptions apply to the dynamic portion of the error file dump, when the error is an MQ error.

See the *WebSphere MQ Application Programming Reference* and the *WebSphere MQ Application Programming Guide* for more information on WebSphere MQ, and WebSphere MQ Integrator calls.

Completion code: (ERR-REC-MQ-COMPCODE):
Data

Indicates the completion code returned on the MQI calls. See the *WebSphere MQ Application Programming Reference* for more information on valid reason codes and values.

Reason code: (ERR-REC-MQ-REASON):
Data

Indicates the reason code qualifying the completion code returned on the MQI call. See the *WebSphere MQ Application Programming Reference* for more information on valid reason codes and values.

Object type: (ERR-REC-MQ-OBJECTTYPE):
Data

Indicates the type of object named in **Object name** field. In the current version of CICS Service Flow Runtime, this field will always indicate Queue (i.e., ERR-REC-MQ-OBJECTTYPE = +1).

Object name : (ERR-REC-MQ-OBJECTNAME):
Data

Indicates the local name of the object as defined on the queue manager identified by **Qmgr name**.

QMgr name : (ERR-REC-MQ-OBJECTQMGRNAME):
Data

Indicates the name of the queue manager on which the **Object name** is defined.

Reslvd Qname : (ERR-REC-MQ-RESOLVEDQNAME):
Data

If there is a value, it indicates the name of the final destination queue, as known to the local queue manager.

Reslvd QMgr : (ERR-REC-MQ-RESOLVEDQMGRNAME):
Data

If there is a value, it indicates the name of the final destination queue manager, as known to the local queue manager.

Dynamic portion of error dump — Link3270 errors: The following field descriptions apply to the dynamic portion of the error file dump, when the error is a Link3270 error.

Service name : (ERR-REC-L3270-SERVICE-NAME):

Data

The service name used in the Link3270 server adapter for facility state file processing. This field can be blank in this dump. If specified, it is used to provide part of the Link3270 State file key if it is desirable to have multiple bridge facilities allocated, remain allocated and in use for a specific userid in adapter server processing. Also if specified, this value is derived from the TYPE = 5 record, PARM01, in the CICS Service Flow Runtime Properties file. The specific TYPE = 5 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump.

System name: (ERR-REC-L3270-REGIONID):
Data

The system name of the CICS server region to which the Link3270 bridge facility is allocated if AOR routing is specified. The AOR routing in-use indicator is derived from the TYPE = 5 record, PARM07, in the CICS Service Flow Runtime Properties file. The specific TYPE = 5 record, that is, the key value, can be determined by the value found in the **Program** field in the static portion of this dump. If AOR routing is not in-use, this field specifies the local system name of the CICS region running CICS Service Flow Runtime where the problem was reported.

CICS Resp: (ERR-REC-L3270-RESP/ERR-REC-LB-RESP) :
Data

This field, if present, indicates the CICS RESP code returned either on the EXEC CICS INQUIRE TRANSACTION command, ERR-REC-L3270-RESP, or on the EXEC CICS LINK command, ERR-REC-LB-RESP. See the *CICS Application Programming Reference* for more information on valid CICS RESP codes and values.

CICS Resp2: (ERR-REC-L3270-RESP2/ERR-REC-LB-RESP2):
Data

This field, if present, indicates the CICS RESP2 code returned either on the EXEC CICS INQUIRE TRANSACTION command, ERR-REC-L3270-RESP, or on the EXEC CICS LINK command, ERR-REC-LB-RESP. See the *CICS Application Programming Reference* for more information on valid CICS RESP2 codes and values.

Field name : ERR-REC-L3270-FIELD-NAME):
Data

The BMS map DFHMDF field name. If present in this dump, a warning was issued whereby either an attempt was made to update a protected field or an invalid attribute was found for this indicated field name.

Attribute: (ERR-REC-L3270-INV-ATT):
Data

If present in this dump, a warning was issued that an invalid attribute was found for the indicated field name. The attribute is dumped in both display and hexadecimal formats.

Linked-to program : (ERR-REC-LB-PROGRAM):

Data

This field, if present, indicates the attempted linked-to program name used in the EXEC CICS LINK command.

Length : (ERR-REC-LB-LENGTH):
Data

This field, if present, indicates the length in bytes of the COMMAREA passed in the failed EXEC CICS LINK command.

Transid : (ERR-REC-BRIH-TRANSACTIONID):
Data

This field, if present, indicates the transaction ID specified on the failed EXEC CICS INQUIRE TRANSACTION command.

Dynamic portion of error dump — Temporary storage queue errors: The following field descriptions apply to the dynamic portion of the error file dump when the error is a temporary storage queue error. Currently CICS Service Flow Runtime uses temporary storage queues (TSQs) for Link3270 server adapter processing only. Additionally, TSQs are only in use when processing adapter services of the *single connector nonpersistent type*.

TSQ Qname: (ERR-REC-TS-NAME):
Data

Indicates the Link3270 Business State TSQ name, (CICS READQ TS QNAME) for which the error occurred.

The 16 byte TSQ QNAME format is:

```
'DFHMA' + 8 byte bridge facility token + 3 byte x 'FFFFFF'
```

CICS resp: (ERR-REC-TS-RESP):
Data

Indicates the CICS RESP code returned for the **TSQ function** performed. See the *CICS Application Programming Reference* for more information on valid CICS RESP codes and values used on indicated EXEC CICS APIs

resp2: (ERR-REC-TS-RESP2):
Data

Data: Indicates the CICS RESP2 code returned for the **TSQ function** performed. See the *CICS Application Programming Reference* for more information on valid CICS RESP2 codes and values used on indicated EXEC CICS APIs.

TSQ function: (ERR-REC-TS-FUNCTION) :
Data

Indicates the function attempted for the defined TSQ that resulted in the error or I/O failure.

Valid values are as follows:

- Inquire start (CICS INQUIRE TSQNAME START AT)
- Inquire next (CICS INQUIRE TSQNAME with NEXT option)

- Inquire end (CICS INQUIRE TSQNAME with END option)
- Read (CICS READQ TS QNAME)
- Write (CICS WRITEQ TS QNAME with MAIN option)
- Rewrite (CICS WRITEQ TS QNAME with REWRITE option)
- Delete (CICS DELETEQ TS QNAME)

See the *CICS Application Programming Reference* and the *CICS System Programming Reference* for more information.

TSQ item: (ERR-REC-TS-ITEM):
Data

Indicates the TSQ item number value (ITEM option) for TSQ used in the function attempted that resulted in the error or I/O failure.

The TSQ item number valid values are +1 and +2 and are defined as follows:

- **Item +1** contains the Link3270 facility business state information used in Link3270 server adapter processing.
- **Item +2** 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.

TSQ length: (ERR-REC-TS-LEN):
Data

Indicates the specified data record length value (LENGTH option) for TSQ used in the function attempted that resulted in the error or I/O failure.

TSQ data: (ERR-REC-TS-DATA):
Data

Indicates the first 112 bytes of TSQ record data used in the function attempted that resulted in the error or I/O failure. This data is dumped in both display and hex formats.

Dynamic portion of error dump — CICS API command errors: The following field descriptions apply to the dynamic portion of the error file dump when the error is a CICS API command error. Each CICS Service Flow Runtime error message corresponds to a specific CICS API command that resulted in failure.

Note: A CICS Service Flow Runtime error message CIA07016E (MAPSET-LOAD-ERRMSG) and CIA07017E (MAP-NOT-FOUND-ERRMSG) may be reported as a CICS API command error. This can occur if in the passthrough XML parse and build program, DFHMAXPI, the indicated BMS map set name could not be loaded into main storage as the result of an EXEC CICS LOAD PROGRAM command. See **Resource name** field description below.

CICS Resp: (ERR-REC-CICS-RESP):
Data

Indicates the CICS RESP code returned for the CICS API command performed. See the *CICS Application Programming Reference* for more information on valid CICS RESP codes and values used on indicated EXEC CICS APIs. The particular

CICS API command can be determined from the reported CICS Service Flow Runtime error message.

CICS Resp2: (ERR-REC-CICS-RESP2):
Data

Indicates the CICS RESP2 code returned for the CICS API command performed. See the *CICS Application Programming Reference* for more information on valid CICS RESP2 codes and values used on indicated EXEC CICS APIs. The particular CICS API command can be determined from the reported CICS Service Flow Runtime error message.

Resource name: (ERR-REC-RESOURCE-NAME):
Data

This is an optional field value supplied only when the error message is CIA07016E (MAPSET-LOAD-ERRMSG).

The **Resource name** will indicate the BMS map set name used in the failed **EXEC CICS LOAD PROGRAM** command. The BMS map set name used is found in the passthrough map header structure (CIA-MAP-HEADER) field (CIAMH-MAPSET-NAME). See "CIA-MAP-HEADER header structure" on page 94 for further information.

Dynamic portion of error dump — XML parse or build errors: The following field descriptions apply to the dynamic portion of the error file dump when the error is an XML parse or build processing error.

Reference the *Enterprise COBOL for z/OS and OS/390 Programming Guide* for XML exception codes and definitions. Also, examine destination **CEEMSG** for messages output by language environment to determine why the XML parse failed.

Currently, the following fields will only be present in the error file dump as the result of an error when CICS Service Flow Runtime run time is processing in passthrough mode.

XML code: (ERR-REC-XML-CODE):
Data

Indicates the XML numeric exception code returned from the parser in special register *XML-CODE*. A value of **-1** before indicates a CICS Service Flow Runtime initiated exception.

XML event: (ERR-REC-XML-EVENT):
Data

Indicates the name of the XML event returned from the parser in special register *XML-EVENT*. If this field indicates *EXCEPTION*, an error in processing the XML document is detected.

For encoding conflict exceptions, which are signaled before parsing begins, fields **XML text** and **XML text length** are either zero-length or contain just the encoding declaration value from the document and length, respectively.

XML text length: (ERR-REC-XML-TEXT-LEN) :

Data

Indicates the actual length of the document text found in special register *XML-TEXT* and not the length of the **XML text** field found in this error file dump.

XML text: (ERR-REC-XML-TEXT):
Data

Indicates the last 160 bytes of document text found in special register *XML-TEXT*. It contains the document text just prior to and including the point where the exception was detected (i.e.,.the part of the document that was parsed prior to and including the point where the exception (the "superfluous data" after the

```
<samdwich>
```

element end tag was detected.

Dynamic portion of error dump — Field validation error: The following field descriptions apply to the dynamic portion of the error file dump when the error is an invalid field value found in the passthrough header structure DFHMAH2 when processing in passthrough mode. This applies when the error message is CIA08005E (VALIDATION-ERRMSG).

Currently, the following fields will only be present in the Error File dump as the result of an error when processing in passthrough mode.

See the field names and definitions in “DFHMAH2 header structure” on page 87 for further information.

Field name: (ERR-REC-FIELD-NAME):
Data

Indicates the passthrough header structure DFHMAH2 field name in error.

Type: (ERR-REC-FIELD-TYPE):
Data

Indicates the field type of the field name in error. This value represents the defined COBOL PICTURE clause of the field name in error.

Valid values are as follows:

- B - Binary fullword
- H - Binary halfword
- X - Alphanumeric display
- 9 - Numeric display

Length: (ERR-REC-FIELD-LEN):
Data

Indicates the defined field length as indicated by the COBOL PICTURE clause of the field name in error. If the field in error is a binary type, this length value indicates the number of bytes of storage occupied not the digit value found in the PICTURE clause.

Field value: (ERR-REC-NUMERIC-VALUE or ERR-REC-ALPHA-VALUE) :

Data

Indicates the actual field value provided in error.

Link3270 message dump format: The remainders of the fields, if present, represent the inbound/outbound Link3270 bridge message field values in CICS Service Flow Runtime.

If **Return code**: is not equal to zero (OK), the Link3270 bridge mechanism has reported an error. This is as the result of one of the following:

- a prior inbound ALLOCATE facility message sent from CICS Service Flow Runtime to the Link3270 bridge mechanism (DFHL3270).
- a prior inbound DELETE facility message sent from CICS Service Flow Runtime to the Link3270 bridge mechanism (DFHL3270).
- an inbound vector message (BRIV) sent from CICS Service Flow Runtime to the Link3270 bridge mechanism (DFHL3270).

Each inbound/outbound vector message structure (BRIV) is preceded by the Link3270 bridge header structure, **BRIH**.

The Link3270 bridge header structure (BRIH) is mapped by copybook DFHBRIHO. Any inbound vector message structure (BRIV) is mapped by DFHBRIIO. Any outbound vector message structure (BRIV) is mapped by DFHBRIOO. These copybooks are CICS Transaction Server 2.2 copybooks and can be found in **cicshlq.SDFHCOB**.

See the *CICS External Interfaces Guide Version 2 Release 2* for more information on the Link3270 bridge mechanism, Link3270 bridge header structure (BRIH), valid return, completion and reason codes and inbound and outbound vector message structures (BRIV).

In describing the format of the Inbound/Outbound message dump, those fields that appear in (**PARENTHESES**) are the corresponding Link3270 bridge header structure (BRIH) fields, CICS Service Flow Runtime Error file record descriptor fields from copybook DFHMARER, and any inbound and/or outbound vector message structure (BRIV) fields from the copybooks DFHBRIIO and DFHBRIOO, respectively.

Please note that group items **(ERR-REC-BRIV-HEADER)** and **(ERR-REC-BRIV-FIELDS)** that are found in DFHMARER are mapped to the appropriate and corresponding inbound and outbound vector message structures (BRIV) when formatting this dump. Also please note, this is a sample dump format. Other inbound/outbound vectors (BRIV) may appear in addition to the SEND MAP and RECEIVE MAP vectors shown here.

Inbound/Outbound message dump format: **BRIH**; common to both inbound/outbound messages:

```
          Structure: (BRIH-STRUCID)
            Version: (BRIH-VERSION)
  Structure length: (BRIH-STRUCLENGTH)
     Facilitytoken: (ERR-REC-BRIH-FACILITYTOKEN)
           Netname: (ERR-REC-BRIH-NETNAME)
          Terminal: (ERR-REC-BRIH-TERMINAL)
     Transactionid: (ERR-REC-BRIH-TRANSACTIONID)
        Datalength: (ERR-REC-BRIH-DATALENGTH)
```

Inbound messages only:

```
   Getwaitinterval:  (ERR-REC-BRIH-GETWAITINTERVAL)
  Facilitykeeptime:  (ERR-REC-BRIH-MAXKEEPTIME)
      Facilitylike:  (ERR-REC-BRIH-FACILITYLIKE)
               AID:  (ERR-REC-BRIH-ATTENTIONID)
         Startcode:  (ERR-REC-BRIH-STARTCODE)
        Cancelcode:  (ERR-REC-BRIH-CANCELCODE)
     ADSDescriptor:  0 (NO)
Conversationaltask:  (ERR-REC-BRIH-CONVTASK)
    Cursorposition:  (ERR-REC-BRIH-CURSORPOS)
```

Outbound/Inbound response messages only:

```
        Return code:  (ERR-REC-BRIH-RETURNCODE)
          Comp code:  (ERR-REC-BRIH-COMPCODE)
        Reason code:  (ERR-REC-BRIH-REASON)
           Function:  (ERR-REC-BRIH-FUNCTION)
          Abendcode:  (ERR-REC-BRIH-ABENDCODE)
      Taskendstatus:  (ERR-REC-BRIH-TASKENDSTATUS)
Remainingdatalength:  (ERR-REC-BRIH-REMAINDATALEN)
  Nexttransactionid:  (ERR-REC-BRIH-NEXTTRANID)
   Nexttranidsource:  (ERR-REC-BRIH-NEXTTRANIDSRC)
        Erroroffset:  (ERR-REC-BRIH-ERROROFFSET)
              Seqno:  (ERR-REC-BRIH-SEQNO)
```

SEND MAP outbound vector: **BRIV**; format is different for inbound and outbound vectors. Only one will appear.

```
BRIV vector length:  (BRIV-OUTPUT-VECTOR-LENGTH)
SM Erase indicator:  (BRIV-SM-ERASE-INDICATOR)
       SM ERASEAUP:  (BRIV-SM-ERASEAUP-INDICATOR)
  SM Free Keyboard:  (BRIV-SM-FREEKB-INDICATOR)
          SM Alarm:  (BRIV-SM-ALARM-INDICATOR)
          SM FRSET:  (BRIV-SM-FRSET-INDICATOR)
           SM Last:  (BRIV-SM-LAST-INDICATOR)
 SM Wait indicator:  (BRIV-SM-WAIT-INDICATOR)
SM cursor position:  (BRIV-SM-CURSOR)
       SM MSR data:  (BRIV-SM-MSR-DATA)
         SM Mapset:  (BRIV-SM-MAPSET)
            SM Map:  (BRIV-SM-MAP)
 SM data indicator:  (BRIV-SM-DATA-INDICATOR)
    SM data length:  (BRIV-SM-DATA-LEN)
    SM data offset:  (BRIV-SM-DATA-OFFSET)
    SM ADSD length:  (BRIV-SM-ADSD-LEN)
    SM ADSD offset:  (BRIV-SM-ADSD-OFFSET)
```

RECEIVE MAP inbound vector:

```
BRIV vector length:  (BRIV-INPUT-VECTOR-LENGTH)
 M Xmit send areas:  (BRIV-RM-TRANSMIT-SEND-AREAS)
         RM Mapset:  (BRIV-RM-MAPSET)
            RM Map:  (BRIV-RM-MAP)
            RM AID:  (BRIV-RM-AID)
RM cursor position:  (BRIV-RM-CPOSN)
    RM data length:  (BRIV-RM-DATA-LEN)
```

As stated, if **Return code**: is not equal to zero (OK), the Link3270 bridge mechanism has reported an error. The following fields could be the cause:

Facilitylike: (ERR-REC-BRIH-FACILITYLIKE)
Data

This field indicates the name of an installed terminal used as a model for the bridge facility allocation in the identified Link3270 server adapter. This is an optional parameter. If no value is specified, the CICS-supplied definition CBRF, is used.

Facilitykeeptime: (ERR-REC-BRIH-MAXKEEPTIME)
Data

This field indicates the maximum timeout value in seconds for an allocated bridge facility. When this timeout value is exceeded, the bridge facility is deleted. This is an optional parameter. If this value is larger than the BRMAXKEEPTIME system initialization parameter value in the AOR (router region), then CICS changes this parameter in the link parameter list, but does not return the reduced value.

Getwaitinterval: (ERR-REC-BRIH-GETWAITINTERVAL)
Data

This field indicates, for the back-end application transaction, the maximum wait interval for message input (in milliseconds) from the indicated Link3270 server adapter. The maximum wait interval value used is the smaller of the BRIH-GETWAITINTERVAL and the RTIMEOUT value of the back-end application transaction.

These fields are set in the Service Flow Modeler as Link3270 node properties. See “Link3270 node properties” on page 280 for information on Link3270 properties.

Dynamic portion of error dump — BTS errors:

The following field descriptions apply to the dynamic portion of the error file dump, when the error is a CICS / BTS error.

You can determine the BTS command performed that resulted in the error or failure by the message found in **Error**. BTS errors do not include BTS data-container errors encountered. See “Dynamic portion of error dump — Data-container errors” on page 293 for descriptions of fields that result from data-container errors.

See the *CICS Business Transaction Services* manual for more information on BTS, and CICS BTS APIs.

CICS Resp : (ERR-REC-BTS-RESP):
Data

Indicates the CICS RESP code returned on the BTS command performed. See the *CICS Application Programming Reference* for more information on valid CICS RESP codes and values.

CICS Resp2 : (ERR-REC-BTS-RESP2):
Data

Indicates the CICS RESP2 code qualifying the CICS RESP returned on the BTS command performed. See the *CICS Business Transaction Services* manual for more information on valid CICS RESP2 codes and values on BTS APIs.

Compstatus : (ERR-REC-BTS-COMPSTATUS):
Data

Indicates the completion status of the activity or process.

Valid values are:

- Abend (ERR-REC-BTS-COMPSTATUS = +900)
- Forced (ERR-REC-BTS-COMPSTATUS = +1013)

- Incomplete (ERR-REC-BTS-COMPSTATUS = +1014)
- Normal (ERR-REC-BTS-COMPSTATUS = +1016)
- None (ERR-REC-BTS-COMPSTATUS = zero)
- UNKNOWN (ERR-REC-BTS-COMPSTATUS = unknown value)

This field will be loaded if the BTS error or failure occurs after the BTS CHECK ACTIVITY or BTS CHECK ACQPROCESS has been performed (i.e., upon return from a child activity or the Root activity of the process).

Mode : (ERR-REC-BTS-MODE):
Data

Indicates the completion status of the activity or process.

Valid values are:
- Active (ERR-REC-BTS-MODE = +181)
- Initial (ERR-REC-BTS-MODE = +789)
- Dormant (ERR-REC-BTS-MODE = +1024)
- Cancelling (ERR-REC-BTS-MODE = +1025)
- Complete (ERR-REC-BTS-MODE = +1026)
- None (ERR-REC-BTS-MODE = zero)
- UNKNOWN (ERR-REC-BTS-MODE = unknown value)

This field will be loaded if the BTS error or failure occurs after the BTS CHECK ACTIVITY or BTS CHECK ACQPROCESS has been performed (i.e., upon return from a child activity or the Root activity of the process).

Suspstatus : (ERR-REC-BTS-SUSPSTATUS):
Data

Indicates the completion status of the activity or process.

Valid values are:
- Suspended (ERR-REC-BTS-SUSPSTATUS = +231)
- Not suspended (ERR-REC-BTS-SUSPSTATUS = +1027)
- None (ERR-REC-BTS-SUSPSTATUS = zero)
- UNKNOWN (ERR-REC-BTS-SUSPSTATUS) = unknown value)

This field will be loaded if the BTS error or failure occurs after the BTS CHECK ACTIVITY or BTS CHECK ACQPROCESS has been performed (i.e., upon return from a child activity or the Root activity of the process).

Program : (ERR-REC-BTS-PROGRAM):
Data

This indicates the target program name expressed either implicitly or explicitly in the execution of a BTS command (i.e., explicitly stated in the PROGRAM parameter on the BTS DEFINE ACTIVITY command or implicitly in a BTS RUN ACTIVITY command).

If the **Program** field in the static portion of this dump indicates an error in the CICS Service Flow Runtime DPL Stub program (DFHMADPL), and you encounter a BTS

error, the program name value should be **DFHMAMGR** indicating the error occurred while trying to perform the following BTS-related tasks:

- Define the process (BTS DEFINE)
- Run the acquired process (BTS RUN)
- Check the status of the process (BTS CHECK)
- Acquire a failed process for purposes of compensation (BTS ACQUIRE)
- Cancel a failed process (BTS CANCEL)

If the **Program** field in the static portion of this dump indicates an error in the CICS Service Flow Runtime Navigation Manager (DFHMAMGR), and you encounter a BTS error, the program name value will be the adapter Navigator program name invoked for the particular request, see **Request field** in the static portion of this dump. The Navigator program name can be found in the TYPE = R record, PARM02, for this request. The error occurred while trying to perform the following BTS-related tasks:

- Define the activity (BTS DEFINE)
- Run the activity (BTS RUN)
- Check the status of the activity (BTS CHECK)

If the **Program** field in the static portion of this dump indicates an error in the adapter Navigator program and you encounter a BTS error, the program name value will be an server adapter name invoked as part of the modeled adapter flow. This value will correspond to the key value of a TYPE = 1, 2, 3, or 4 record in the CICS Service Flow Runtime Properties file. The error occurred while trying to perform the following BTS-related tasks:

- Define the activity (BTS DEFINE)
- Run the activity (BTS RUN)
- Check the status of the activity (BTS CHECK)

Transid : (ERR-REC-BTS-TRANSACTION):
Data

This indicates the target transaction expressed either implicitly or explicitly on or in the execution of a BTS command i.e., explicitly stated in the TRANSID parameter on the BTS DEFINE ACTIVITY command or implicitly in a BTS RUN ACTIVITY command).

If the **Program** field in the static portion of this dump indicates an error in the CICS Service Flow RuntimeDPL Stub (DFHMADPL) and you encounter a BTS error, the transaction Id value should be **CMAM** indicating the error occurred while trying to perform the following BTS-related tasks:

- Define the process (BTS DEFINE)
- Run the acquired process (BTS RUN)
- Check the status of the process (BTS CHECK)
- Acquire a failed process for purposes of compensation (BTS ACQUIRE)
- Cancel a failed process (BTS CANCEL)

If the **Program** field in the static portion of this dump indicates an error in the CICS Service Flow Runtime Navigation Manager (DFHMAMGR), and you encounter a BTS error, the transaction Id value will be the adapter Navigator transaction invoked for the particular request, see **Request** field in the static portion of this dump. The

Navigator transaction Id can be found in the TYPE = R record, PARM03, for this request. The error occurred while trying to perform the following BTS-related tasks:

- Define the activity (BTS DEFINE)
- Run the activity (BTS RUN)
- Check the status of the activity (BTS CHECK)

If the **Program** field in the static portion of this dump indicates an error in the adapter Navigator program and you encounter a BTS error, the transaction Id value will be an server adapter transaction Id invoked as part of the modeled adapter flow. The error occurred while trying to perform the following BTS-related tasks:

- Define the activity (BTS DEFINE)
- Run the activity (BTS RUN)
- Check the status of the activity (BTS CHECK)

Activity : (ERR-REC-BTS-ACTIVITY):
Data

This indicates the target activity name expressed either implicitly or explicitly in the execution of a BTS command (i.e., explicitly stated in the ACTIVITY parameter on the BTS DEFINE ACTIVITY command or implicitly in a BTS RUN ACTIVITY command). In the current version of CICS Service Flow Runtime, this value will always equal the value found in the BTS error **Program** field.

Dynamic portion of error dump — Abend errors: The following field descriptions apply to the dynamic portion of the error file dump, when the error is a CICS abend error.

See the *CICS Messages and Codes* manual for valid abend codes, values and user actions. Also see the *CICS Application Programming Reference* for information regarding EXEC CICS HANDLE ABEND, EXEC CICS ASSIGN and EXEC CICS ABEND commands.

Abcode : (ERR-REC-ABCODE):
Data

Indicates the CICS abend code returned in the HANDLE routine as the result of an EXEC CICS ASSIGN command. If it indicates a value of **CIA**, this is an abend code forced by CICS Service Flow Runtime run time server. It occurs when processing in Synchronous Rollback mode and an error or failure occurs during server adapter processing. The request processing mode is indicated on a TYPE = R record, PARM01, for the request being processed, see **Request** field in the static portion of this dump.

Abdump : (ERR-REC-ABDUMP):
Data

Indicates if a CICS dump has been produced as the result of the abend. The value is returned in the HANDLE routine as the result of an EXEC CICS ASSIGN command.

Valid values are:

- DUMP (ERR-REC-ABDUMP = X'FF')
- NODUMP (ERR-REC-ABDUMP = X'00')

Abprogram : (ERR-REC-ABPROGRAM):
Data

The CICS program name returned in the HANDLE routine as the result of an EXEC CICS ASSIGN command indicating the failed program for the abend. If the abend originally occurred in a DPL target and server program running in a remote system, this value indicates the DPL target and server program name. This field is set to binary zeros if the failing program could not be determined at the time of the abend.

SYSID : (ERR-REC-SYSID):
Data

The local 4 byte CICS name returned in the HANDLE routine as the result of an EXEC CICS ASSIGN command. If the abend originally occurred in a DPL target and server program running in a remote system, this value indicates the value used in the SYSID parameter on the EXEC CICS LINK command.

CICS Resp: (ERR-REC-ASSIGN-RESP):
Data

Indicates the CICS RESP code returned on the EXEC CICS ASSIGN command. See the *CICS Application Programming Reference* for more information on valid CICS RESP codes and values.

CICS Resp2 : (ERR-REC-ASSIGN-RESP2):
Data

Indicates the CICS RESP2 code qualifying the CICS RESP returned on the EXEC CICS ASSIGN command. See the *CICS Application Programming Reference* for more information on valid CICS RESP2 codes and values.

Dynamic portion of error dump - BIDI transformation errors:

The following field descriptions apply to the dynamic portion of the error file dump, when the error is a BIDI transformation error. When a BIDI transformation error occurs, CICS Service Flow Runtime records all parameter data used to call the BIDI transformation routine. Please note that for all BIDI transformation response codes other than 5, this information is only of use to the IBM Service team in helping to diagnose the problem.

Response (ERR-REC-BIDITRN-RESP):
Data

Indicates the response code from the BIDI transformation module. A response code of 5 indicates an invalid attribute string - the reason code (ERR-REC-BIDI-REAS) indicates whether it is a duplicate reference (value 1), has no Text Type attribute provided (value 2) or has no Orientation attribute provided (value 3). The other response codes indicate that the systems functions used to support the BIDI transformation have failed. The reason code provides the error code returned by the failing system function. These are provided for diagnostic purposes for the IBM Service Team.

Reason (ERR-REC-BIDITRN-REAS):

Data

Indicates the reason code from the BIDI transformation module. If the response code (ERR-REC-BIDI-RESP) is 5, then this value will indicate whether it is a duplicate reference (value 1), has no Text Type attribute provided (value 2) or has no Orientation attribute provided (value 3). 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 are provided for diagnostic purposes for the IBM Service Team.

Module called (ERR-REC-BIDITRN-MODULE):
Data

Indicates the name of the module called for BIDI transformation. This will either be FEJBDTRN or FEJBDTRE depending on the options chosen when the Adapter Service was modeled in Service Flow Modeler.

INATTR (ERR-REC-BIDITRN-INATTR):
Data

Contains the first 20 bytes of the attribute string used as input to the routine.

INATTR length (ERR-REC-BIDITRN-INATTRLEN):
Data

The length, in bytes, of the attribute string used for input.

OUTATTR (ERR-REC-BIDITRN-OUTATTR):
Data

Contains the first 20 bytes of the attribute string returned by the BIDI transformation routine. This shows the transformation that took place.

OUTATTR Length (ERR-REC-BIDITRN-OUTATTRLEN):
Data

The length, in bytes, of the attribute string returned by the BIDI transformation routine.

Code page (ERR-REC-BIDITRN-CODEPAGE):
Data

The code page, represented as a text string, of the text to be transformed.

Code page length (ERR-REC-BIDITRN-CODEPAGELEN):
Data

The length, in bytes, of the code page.

Data (ERR-REC-BIDITRN-DATA):
Data

The first 100 bytes of the BIDI data to be transformed.

Data length (ERR-REC-BIDITRN-DATALEN):

Data

The length of the data to be transformed by the BIDI transformation routine.

Dynamic portion of error dump — Application errors: Indicates that an application processing error has occurred. This error type is associated with error messages CIA080xxE.

See “Additional error messages” on page 240 for a description of these error types. There is no error detail for a CIA08002E. The error detail for error message CIA08001E indicates the invoking program as returned from the INVOKINGPROG parameter of the EXEC CICS ASSIGN command, see the *CICS Application Programming Reference*.

Vector file dump

This 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 adapter request 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 a TYPE=R record in the properties file and is derived from the DFHMAH message structure in the request message.
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. It is derived from the TYPE=5 record and **PARM01** parameter in the properties file. The specific record can be determined by the value found in the Program field of this dump.
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. It is derived from the DFHMAH-PROCESSTYPE field in the DFHMAH message header.
Process	The process name of the adapter request processing instance under which the Link3270 server adapter was running. It is derived from the DFHMAH-PROCESSNAME field in the DFHMAH message header.
Date	The date that the records were written to the vector log file.

Field name	Description
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 only used 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 will appear 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). This is provided when you run full vector logging (MP-BR-VECTOR-LOGGING = +1; see “Link3270 node properties” on page 280). 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 or text data 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 mechanism.

Reading the Vector log file dump

The following information describes the fields on the formatted CICS Service Flow Runtime Vector file dump.

In describing the contents of the formatted vector log file dump, those fields that appear in **bold** are the fields on the formatted vector log file dump. The fields that appear in **(PARENTHESES)** are the corresponding CICS Service Flow Runtime Link3270 Vector log record field names.

Note: When using full vector logging (MP-BR-VECTOR-LOGGING = +1; see “Link3270 node properties” on page 280), the vector data that will display as contents in the vector dump is the accumulated map buffer or text data that is returned to the generated Link3270 navigator. The buffer contents in the vector dump **may not always contain the individual vector data** returned from the Link3270 bridge mechanism.

The CICS Service Flow Runtime Link3270 Vector log file record descriptor copybook is DFHMARLV.

The remainder of the fields, if present, represent the inbound/outbound Link3270 bridge message field values in CICS Service Flow Runtime. If **Return code**: is not equal to zero (OK), the Link3270 bridge mechanism has reported an error. This is as the result of a prior inbound ALLOCATE facility message, DELETE facility message or inbound vector message (BRIV) sent from CICS Service Flow Runtime to the Link3270 bridge mechanism (DFHL3270).

Each inbound/outbound vector message structure (BRIV) is preceded by the Link3270 bridge header structure, BRIH.

The Link3270 bridge header structure (BRIH) is mapped by copybook DFHBRIHO. Any inbound vector message structure (BRIV) is mapped by DFHBRIIO. Any outbound vector message structure (BRIV) is mapped by DFHBRIOO. These copybooks are CICS Transaction Server 2.2 copybooks and can be found in **cicshlq.SDFHCOB**.

See the *CICS External Interfaces Guide Version 2 Release 2* for more information on the Link3270 bridge mechanism, Link3270 bridge header structure (BRIH), valid return, completion and reason codes and inbound and outbound vector message structures (BRIV).

In describing the contents of the formatted vector log file dump, the fields that appear in (PARENTHESES) are the corresponding Link3270 bridge header structure (BRIH) and any inbound and/or outbound vector message structures (BRIV) from the copybooks DFHBRIHO, DFHBRIIO and DFHBRIOO respectively. These copybooks are CICS Transaction Server 2.2 copybooks and can be found in **cicshlq.SDFHCOB**.

Please note that group items **(LV-BRIH-STRUCTURE)**, **(LV-BRIV-HEADER)** and **(LV-BRIV-FIELDS)** found in DFHMARLV are mapped to the Link3270 bridge header structure (BRIH) and the appropriate and corresponding inbound and outbound vector message structures (BRIV) when formatting this dump. Also please note, this is a sample dump format. Other inbound/outbound vectors (BRIV) may appear in addition to the SEND MAP and RECEIVE MAP vectors shown here.

Conversion templates

You might need to add one or both of the following conversion templates based upon how you are initiating CICS Service Flow Runtime adapter services.

If you are using standard CICS conversion, these templates can be used and added as required to the CICS conversion table, DFHCNV, for code page conversion. For information about implementing data conversion, see “Setting up data conversion” on page 29.

“Data conversion” on page 74 describes how the CICS Service Flow Runtime performs data conversion.

The following conversion templates must also be present in the custom CICS Service Flow Runtime conversion table to support the CICS Web Interface.

```
DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=DFHWBHH,USREXIT=NO,
   SRVERCP=037,CLINTCP=8859-1
DFHCNV TYPE=SELECT,OPTION=DEFAULT
DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=32767,
   LAST=YES

DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=DFHWBUD,USREXIT=NO,
   CLINTCP=8859-1,SRVERCP=037
DFHCNV TYPE=SELECT,OPTION=DEFAULT
DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=32767,
   LAST=YES
```

All conversion templates must be found within the following macros:

- DFHCNV TYPE=INITIAL (defines the beginning of the conversion table)

- DFHCNV TYPE=FINAL (defines the end of the conversion table)

DFHMADPL conversion template

If you are not processing in passthrough mode, use the following conversion template to create a load module in the required CICS load library.

Note: The asterisks are required in column 72 as continuation characters.

```
DFHCNV TYPE=INITIAL
DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=DFHMADPL,USREXIT=NO,      *
     SRVERCP=037,CLINTCP=8859-1
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
```

DFHMADPP conversion template

If you are processing in passthrough mode, use the following conversion template to create a load module in the required CICS load library.

Note: The asterisks are required in column 72 as continuation characters.

```
DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=DFHMADPP,USREXIT=NO,   *
     SRVERCP=037,CLINTCP=8859-1
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=4
DFHCNV TYPE=FIELD,OFFSET=388,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=392,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=396,DATATYP=CHARACTER,DATALEN=16
DFHCNV TYPE=FIELD,OFFSET=412,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=416,DATATYP=CHARACTER,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=420,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=424,DATATYP=CHARACTER,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=428,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=432,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=436,DATATYP=CHARACTER,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=440,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=444,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=448,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=452,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=456,DATATYP=CHARACTER,DATALEN=40
DFHCNV TYPE=FIELD,OFFSET=496,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=500,DATATYP=NUMERIC,DATALEN=4
DFHCNV TYPE=FIELD,OFFSET=504,DATATYP=CHARACTER,DATALEN=44
DFHCNV TYPE=FIELD,OFFSET=548,DATATYP=CHARACTER,DATALEN=32212, *
     LAST=YES
```

IVP sample listing

This section lists the samples (programs, copybooks, maps, JCL and resource definitions) that the IVP uses. All samples are located in the .SCIZSAMP runtime library.

The IVP back-end transactions that are provided

The following table lists the programs and jobs provided to simulate back-end transactions during the IVP.

Table 27. IVP back-end transactions — programs

Description	Name	Tran ID
Back-end 3270 Customer Information Maintenance test transaction program	DFHMABP1	CMAV
Back-end 3270 Account Information Maintenance test transaction program	DFHMABP2	CMAW
Back-end 3270 Customer Name Search test transaction program	DFHMABP3	CMAX
Back-end DPL Customer Information Maintenance test transaction program	DFHMABP4	none
Back-end DPL Account Information Maintenance test transaction program	DFHMABP5	none
Back-end MQ Customer Information Maintenance test transaction program	DFHMABP6	CMAY
Back-end MQ Account Information Maintenance test transaction program	DFHMABP7	CMAZ
Back-end VSAM file initialization program	DFHMABP8	This is a COBOL batch program.

Table 28. IVP back-end transactions — JCL

Description	Name
Back-end Customer Information file IDCAMS define	DFHMABD1
Back-end Customer Information file IDCAMS delete	DFHMABD2
Back-end Account Information file IDCAMS define	DFHMABD3
Back-end Account Information file IDCAMS delete	DFHMABD4
Back-end Customer Name file IDCAMS define	DFHMABD5
Back-end Customer Name file IDCAMS delete	DFHMABD6
JCL to delete, define and initialize back-end VSAM files	DFHMABD7

Table 29. IVP back-end transactions — copybooks and mapsets

Description	Name
Copybook used in back-end test transactions	DFHMABC1
Copybook used in back-end test transactions	DFHMABC2
Copybook used in back-end test transactions	DFHMABC3
Copybook used in back-end test transactions	DFHMABC4
Mapset for back-end test transactions	DFHMAB1

The IVP adapter programs that are provided

The following sample programs simulate deployed adapters. These are the programs that run in the CICS region when you execute the IVP.

Table 30. IVP adapter programs

Description	Name	Tran ID
IVP Navigator program	DFHMAIP1	CMA5
IVP DPL program	DFHMAIP2	CMA6
IVP MQ Put program	DFHMAIP3	CMA7
IVP MQ Get program	DFHMAIP4	CMA8
IVP FEPI Navigator program	DFHMAIP5	CMA9
IVP Link3270 Navigator program	DFHMAIP6	CMA0

Table 31. IVP adapter copybooks

Description	Name
Copybook used in IVP programs	DFHMAIC2
Copybook used in IVP programs	DFHMAIC3
Copybook used in IVP programs	DFHMAIC4
Copybook used in IVP programs	DFHMAIC5
Copybook used in IVP programs	DFHMAIC6
Copybook used in IVP programs	DFHMAIC7
Copybook used in IVP programs	DFHBRIHO This is a CICS copybook, located in hlcicsq.SDFHCOB

Table 31. IVP adapter copybooks (continued)

Description	Name
Copybook used in IVP programs	DFHBRIIO This is a CICS copybook, located in hlcicsq.SDFHCOB
Copybook used in IVP programs	DFHBRIOO This is a CICS copybook, located in hlcicsq.SDFHCOB

Table 32. IVP adapter programs resource definition JCL

Description	Name
IVP CICS resource definitions	DFHMAID1
IVP WebSphere MQ resource definitions	DFHMAID2
IVP Properties file update JCL	DFHMAID3
IVP Link3270 Repository file update JCL	DFHMAID4

The IVP Simulator programs that are provided

The following Simulator programs, copybooks and JCL are provided to enable you to run the IVP.

Table 33. IVP Simulator programs to run the IVP

Description	Name	Tran ID
Simulator program 1	DFHMASP1	CMA1
Simulator program 2	DFHMASP2	CMA2
Simulator program 3	DFHMASP3	CMA3
Simulator program 4	DFHMASP4	CMA4
Simulator program 5	DFHMASP5	none
Simulator program 6	DFHMASP6	none

Table 34. IVP Simulator copybooks and mapsets

Description	Name
Copybook used in the Simulator	DFHMASC1
Copybook used in the Simulator	DFHMASC2
Copybook used in the Simulator	DFHMASC3
Copybook used in the Simulator	DFHMASC4
Copybook used in the Simulator	DFHMASC5
Mapset used in the Simulator	DFHMAS1
Mapset used in the Simulator	DFHMAS2
Mapset used in the Simulator	DFHMAS3
Mapset used in the Simulator	DFHMAS4

Table 35. IVP Simulator JCL

Description	Name
Simulator Symbolic VSAM file alternate index build JCL	DFHMASDA
Simulator Request VSAM file alternate index build JCL	DFHMASDB
Simulator Request file IDCAMS define	DFHMASD1
Simulator Request file IDCAMS delete	DFHMASD2
Simulator Symbolic file IDCAMS define	DFHMASD3
Simulator Symbolic file IDCAMS delete	DFHMASD4
Simulator Request VSAM file alternate index definition JCL	DFHMASD5
Simulator Symbolic VSAM file alternate index definition JCL	DFHMASD6
JCL to delete, define, initialize Simulator VSAM files	DFHMASD7

The IVP sample JCL and procedures that are provided

The sample jobs in the following table are used to compile the IVP Simulator and IVP back-end transactions.

Table 36. IVP JCL and PROC

Description	Name
JCL to assemble CICS mapsets	DFHMAIAJ
JCL to compile CICS programs	DFHMAICJ
JCL to compile COBOL (batch) programs	DFHMAIBJ
Procedure to assemble CICS mapsets	DFHMAIAP
Procedure to compile CICS programs. This PROC can also be used in deployed server adapter program compilations.	DFHMAXCP
Procedure to compile COBOL batch programs	DFHMAIBP

Test data for IVP back-end transactions

The following sample test data is added by program DFHMABP8, which is the back-end VSAM file initialization program.

Data added to the Customer Information file: DFHMABCF

The file is mapped by copybook DFHMABC2 in .SCIZSAMP.

```
MOVE '10000'              TO CUSTOMER-NO.
MOVE 'ONE'                TO LAST-NAME.
MOVE 'CUSTOMER'           TO FIRST-NAME.
MOVE '1 MAIN STREET'      TO STREET.
MOVE 'NEWPORT'            TO CITY.
MOVE 'RI'                 TO STATE.
MOVE '02840'              TO ZIP-CODE.
MOVE '401'                TO AREA-CODE.
MOVE '555'                TO EXCHANGE.
MOVE '1212'               TO PHONE-NO.
MOVE 'N'                  TO MARRIED.
MOVE 'IBM'                TO EMPLOYER.
MOVE '3 SILVA LANE'       TO EMP-ADDRESS.
MOVE 'MIDDLETOWN'         TO EMP-CITY.
MOVE 'RI'                 TO EMP-STATE.
MOVE '02842'              TO EMP-ZIP-CODE.
MOVE '401'                TO EMP-AREA-CODE.
MOVE '555'                TO EMP-EXCHANGE.
```

```
MOVE '9999'              TO EMP-PHONE-NO.
MOVE 'CHECKING'          TO ACCOUNT-TYPE-1.
MOVE '1111111'           TO ACCOUNT-NUMBER-1.
```

Data added to the Customer Name (Phonetic) file: DFHMABNF

The file is mapped by copybook DFHMABC4 in .SCIZSAMP

```
MOVE '10000'             TO PHONETIC-ACCT-NO.
MOVE 'ONE'               TO PHONE-LAST-NAME.
MOVE 'CUSTOMER'          TO PHONE-FIRST-NAME.

Data added to the Account Information file:  DFHMABAF
File is mapped by copybook DFHMABC3  in  .SCIZSAMP.

MOVE '10000'             TO ACCOUNT-CUST-NO.
MOVE '1111111'           TO ACCOUNT-NO.
MOVE 'ONE'               TO ACCOUNT-LAST-NAME.
MOVE 'CUSTOMER'          TO ACCOUNT-FIRST-NAME.
MOVE 'Y'                 TO ACCOUNT-OVERDRAFT-PROT
MOVE 'CHECKING'          TO ACCOUNT-TYPE.
MOVE 150000              TO ACCOUNT-BALANCE
```

XML message formats

The following samples are of the CICS Service Flow Runtime message structure in XML, and of the passthrough application data in XML format.

XML message formats for non-passthrough

The following XML message formats are samples for non-passthrough requests.

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 is a sample of the request message (non-passthrough) in XML. For a description of the fields in the CICS Service Flow Runtime Message Headers (DFHMAH and DFHMAHV2) see “DFHMAH field definitions” on page 81.

The following sample message is the XML version of the request message used to execute the CICS Service Flow Runtime server run-time installation verification procedure (IVP).

The value specified in the **dfhmah_datalength** tag 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>0</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>
```

XML message formats for passthrough requests

The following XML message formats are samples for passthrough requests.

Passthrough messages with application data in XML

1. Send an initial transaction to the CICS Service Flow Runtime:

```
<?xml version="1.0" encoding="UTF-8"?>
<screen>
   <input>CMAV<input>
<screen>
```

2. Receive the resulting initial screen from the CICS Service Flow Runtime:

```
<?xml version="1.0" encoding="UTF-8"?>
      <screen attentionid="enter" maps=1 >
        <map name="MAPA" mapset="DFHMAB1" rows="24" columns="80" fields="2">
             <field name="AR03C18" ml="5" pa="u" ia="n" ha="u"/>
             <field name="AR22C70" ml="1" pa="u" ia="n" ha="u"/>
         <map>
     <screen>
```

3. Send the screen containing input fields to the CICS Service Flow Runtime:

```
<?xml version="1.0" encoding="UTF-8"?>
      <screen attentionid="enter" maps=1 >
        <map name="MAPA" mapset="DFHMAB1" rows="24" columns="80" fields="2">
             <field name="AR03C18">10000<field>
             <field name="AR22C70">1<field>
         <map>
     <screen>
```

4. Receive the screen with data from the CICS Service Flow Runtime:

```
<?xml version="1.0" encoding="UTF-8"?>
      <screen attentionid="enter" maps=1 >
        <map name="MAPA" mapset="DFHMAB1" rows="24" columns="80" fields="15">
            <field name="AR03C18" ml="5"  pa="p" ia="n">10000<field>
            <field name="AR04C18" ml="30" pa="p" ia="n">Customer One<field>
            <field name="AR05C18" ml="35" pa="p" ia="n">1 Elm St<field>
            <field name="AR06C18" ml="20" pa="p" ia="n">Newport<field>
            <field name="AR06C48" ml="2"  pa="p" ia="n">RI<field>
                 .
             .
             .
         <map>
     <screen>
```

XSD for passthrough request message

```
<?xml version="1.0" encoding="UTF-8" ?>
- <schema targetNamespace="http://www.DFHMAXMI.com/schemas/DFHMAXMIInterface"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:scr="http://www.DFHMAXMI.com/schemas/DFHMAXMIInterface">
  - <annotation>
  <documentation>*****************************************************************************
   The format of each request/response begins with a main session header (DFHMAHV). For a
   passthrough request/response a Pass Through header (DFHMAH2V) immediately follows DFHMAHV.
   This XML passthrough schema defines request/response elements that follow the passthrough header.
  </documentation>

  <documentation>
   Each XML passthrough requst/response contains a screen header that immediately
   follows DFHMAH2V. See "CIA-SCREEN-HEADER header structure" on page 92
   for detailed description of this header. The screen header contains the attentionID and the count of
   the number of maps, input fields, and text fields contained in the screen. Input fields and text fields
   do not correspond to the fields contained within a map.
  </documentation>
  <documentation>
   Request elements (screen, map, input):
  </documentation>
  <documentation>
   An unformatted or unmapped request is used to begin a new session. In this case the "screen"
   element may contain a single "input" element. The value of the "input" element is the input text,
   beginning with the 4 character transaction ID. Irregardless of whether an input element is present
```

```
 the target 3270 application will be run using the value contained in the passthrough request header
 "dfhmah2_transid" element.
</documentation>
<documentation>
 A mapped request is used to continue an existing session. In this case the "screen" element will
 contain a single "map" element.
</documentation>
<documentation>
 A "text" element is not valid for either a mapped or unmapped input request.
</documentation>
<documentation>
 Response elements (screen, map, text):
</documentation>
<documentation>
 For a response the "screen" element may contain both "text" or "map" elements in any combination.
 A "map" element contains 1 or more "field" elements that correspond to named fields in a BMS map.
</documentation>
<documentation>
 The value of a "text" element contains text from a SEND TEXT command or from a SEND command that
 contains a single line of text.
</documentation>
<documentation>
 An "input" element is not valid in a response. ************************************************
</documentation>
</annotation>
  <element name="screen">
      <complexType>
       <sequence>
              <element maxOccurs="unbounded" minOccurs="0" ref="scr:map"/>
              <choice>
                  <element maxOccurs="unbounded" minOccurs="0" ref="scr:text"/>
                  <element maxOccurs="1" minOccurs="0" ref="scr:input"/>
              </choice>
          </sequence>
          <attribute name="attentionid" use="required">
           <annotation>
            <documentation>
             attentionid= key pressed to submit the input screen
            </documentation>
           </annotation>
           <simpleType>
            <restriction base="string">
                   <enumeration value="enter"/>
                      <enumeration value="clear"/>
                      <enumeration value="pf1"/>
                      <enumeration value="pf2"/>
                      <enumeration value="pf3"/>
                      <enumeration value="pf4"/>
                      <enumeration value="pf5"/>
                      <enumeration value="pf6"/>
                      <enumeration value="pf7"/>
                      <enumeration value="pf8"/>
                      <enumeration value="pf9"/>
                      <enumeration value="pf10"/>
                      <enumeration value="pf11"/>
                      <enumeration value="pf12"/>
                      <enumeration value="pf13"/>
                      <enumeration value="pf14"/>
                      <enumeration value="pf15"/>
                      <enumeration value="pf16"/>
                      <enumeration value="pf17"/>
                      <enumeration value="pf18"/>
                      <enumeration value="pf19"/>
                      <enumeration value="pf20"/>
                      <enumeration value="pf21"/>
                      <enumeration value="pf22"/>
                      <enumeration value="pf23"/>
```

```
                    <enumeration value="pf24"/>
                    <enumeration value="pa1"/>
                    <enumeration value="pa2"/>
                    <enumeration value="pa3"/>
                    <enumeration value="pen"/>
                    <enumeration value="msre"/>
                    <enumeration value="opid"/>
                    <enumeration value="trig"/>
                    <maxLength value="5"/>
                </restriction>
               </simpleType>
            </attribute>
            <attribute name="maps" use="required">
             <annotation>
              <documentation>
               maps= number of maps, including input and text fields
              </documentation>
             </annotation>
             <simpleType>
                   <restriction base="integer">
                       <minInclusive value="0"/>
                       <totalDigits value="4"/>
                   </restriction>
               </simpleType>
            </attribute>
        </complexType>
    </element>
    <element name="map">
        <complexType>
            <sequence>
                <element maxOccurs="unbounded" minOccurs="1" ref="scr:field"/>
            </sequence>
            <attribute name="name" use="required">
             <annotation>
              <documentation>
               name= BMS map name
              </documentation>
             </annotation>
             <simpleType>
                   <restriction base="string">
                       <maxLength value="7"/>
                   </restriction>
               </simpleType>
            </attribute>
            <attribute name="mapset" use="required">
             <annotation>
              <documentation>
               mapset= BMS mapset name
              </documentation>
             </annotation>
             <simpleType>
                   <restriction base="string">
                       <maxLength value="7"/>
                   </restriction>
               </simpleType>
            </attribute>
            <attribute name="rows" use="optional">
             <annotation>
              <documentation>
               rows= number of map rows
              </documentation>
             </annotation>
             <simpleType>
                   <restriction base="integer">
                       <minInclusive value="12"/>
                       <maxInclusive value="43"/>
                       <totalDigits value="2"/>
```

```
                </restriction>
            </simpleType>
        </attribute>
        <attribute name="columns" use="optional">
         <annotation>
          <documentation>
           columns= number of BMS map columns
          </documentation>
         </annotation>
 <simpleType>
                <restriction base="integer">
                    <minInclusive value="40"/>
                    <maxInclusive value="132"/>
                    <totalDigits value="3"/>
                </restriction>
            </simpleType>
        </attribute>
        <attribute name="fields" use="optional">
         <annotation>
          <documentation>
           fields= number of map fields
          </documentation>
         </annotation>
         <simpleType>
                <restriction base="integer">
                    <minInclusive value="1"/>
                    <totalDigits value="3"/>
                </restriction>
            </simpleType>
        </attribute>
        <attribute name="focus" use="optional">
         <annotation>
          <documentation>
           focus= map field name where the cursor
           is initially positioned
          </documentation>
         </annotation>
         <simpleType>
                <restriction base="string">
                    <maxLength value="32"/>
                </restriction>
            </simpleType>
        </attribute>
    </complexType>
</element>
<element name="field">
    <complexType>
     <annotation>
          <documentation>
           field= BMS map field
          </documentation>
         </annotation>
        <simpleContent>
            <extension base="string">
                <attribute name="name" use="required">
                 <simpleType>
            <annotation>
             <documentation>
              name= BMS field name
             </documentation>
            </annotation>
                  <restriction base="string">
                      <maxLength value="30"/>
                  </restriction>
              </simpleType>
          </attribute>
                <attribute name="ml" use="required">
```

```
                <simpleType>
                 <annotation>
           <documentation>
            ml= maximum length
           </documentation>
          </annotation>
                <restriction base="integer">
                    <minInclusive value="1"/>
                    <maxInclusive value="255"/>
                    <totalDigits value="3"/>
                </restriction>
            </simpleType>
</attribute>
              <attribute name="pa" use="required">
                  <simpleType>
                   <annotation>
           <documentation>
            pa= protect attribute:
            p= protect,
            u= unprotect
           </documentation>
          </annotation>
                      <restriction base="string">
                          <enumeration value="p"/>
                          <enumeration value="u"/>
                          <maxLength value="1"/>
                      </restriction>
                  </simpleType>
              </attribute>
              <attribute name="ia" use="required">
                  <simpleType>
                   <annotation>
           <documentation>
            ia= intensity attribute:
            n= normal intensity,
            b= bright intensity,
            d= dark or non-display intensity
           </documentation>
          </annotation>
                      <restriction base="string">
                          <enumeration value="n"/>
                          <enumeration value="b"/>
                          <enumeration value="d"/>
                          <maxLength value="1"/>
                      </restriction>
                  </simpleType>
              </attribute>
              <attribute name="fa" use="optional">
                  <simpleType>
                   <annotation>
           <documentation>
            fa= FSET attribute:
            y= FSET on or modified data tag turned on
           </documentation>
          </annotation>
                      <restriction base="string">
                          <enumeration value="y"/>
                          <maxLength value="1"/>
                      </restriction>
                  </simpleType>
              </attribute>
              <attribute name="da" use="optional">
                  <simpleType>
                   <annotation>
           <documentation>
            da= selector pen detectable attribute:
            y= detectable turned on
```

```
 </documentation>
</annotation>
            <restriction base="string">
                <enumeration value="y"/>
                <maxLength value="1"/>
            </restriction>
        </simpleType>
    </attribute>
    <attribute name="ca" use="optional">
        <simpleType>
         <annotation>
 <documentation>
  ca= color attribute:
  b= blue,
  r= red,
  g= green,
  t= turqoise,
  y= yellow,
  p= pink,
  n= neutral
 </documentation>
</annotation>
            <restriction base="string">
                <enumeration value="b"/>
                <enumeration value="r"/>
                <enumeration value="g"/>
                <enumeration value="t"/>
                <enumeration value="y"/>
                <enumeration value="p"/>
                <enumeration value="n"/>
                <maxLength value="1"/>
            </restriction>
        </simpleType>
    </attribute>
    <attribute name="ha" use="optional">
        <simpleType>
         <annotation>
 <documentation>
  ha= highlight attribute:
  b= blink,
  u= underline,
  r= reverse video
 </documentation>
</annotation>
            <restriction base="string">
                <enumeration value="b"/>
                <enumeration value="u"/>
                <enumeration value="r"/>
                <maxLength value="1"/>
            </restriction>
        </simpleType>
    </attribute>
    <attribute name="va" use="optional">
        <simpleType>
         <annotation>
 <documentation>
  va= Validation attribute:
  f= must fill,
  e= must enter,
  t= trigger
    </documentation>
</annotation>
            <restriction base="string">
                <enumeration value="f"/>
                <enumeration value="e"/>
                <enumeration value="t"/>
                <maxLength value="1"/>
```

```
                    </restriction>
                </simpleType>
            </attribute>
            <attribute name="ps" use="optional">
             <simpleType>
              <annotation>
           <documentation>
            ps= programmed symbols
           </documentation>
          </annotation>
          <restriction base="string">
                        <maxLength value="1"/>
                    </restriction>
         </simpleType>
        </attribute>
              <attribute name="so" use="optional">
                  <simpleType>
                   <annotation>
           <documentation>
            so= shift in / shift out:
            y= yes,
            n= no
           </documentation>
          </annotation>
                    <restriction base="string">
                        <enumeration value="y"/>
                        <enumeration value="n"/>
                        <maxLength value="1"/>
                    </restriction>
                </simpleType>
            </attribute>
            <attribute name="oi" use="optional">
             <simpleType>
              <annotation>
           <documentation>
            oi= occurs index
           </documentation>
          </annotation>
                <restriction base="integer">
                   <minInclusive value="1"/>
                   <totalDigits value="3"/>
               </restriction>
            </simpleType>
          </attribute>
           </extension>
        </simpleContent>
    </complexType>
</element>
<element name="text">
 <simpleType>
  <annotation>
         <documentation>
          text= from SEND TEXT or SEND command
         </documentation>
        </annotation>
        <restriction base="string">
            <maxLength value="255"/>
        </restriction>
    </simpleType>
</element>
<element name="input">
 <simpleType>
  <annotation>
         <documentation>
          input= text entered on unformatted or unmapped screen
         </documentation>
        </annotation>
```

```
            <restriction base="string">
                <maxLength value="132"/>
            </restriction>
        </simpleType>
    </element>
</schema>
```

Application data in XML format

The following example shows the application data portion of a passthrough request message in XML.

Note: The CICS Service Flow Runtime supports passthrough processing for Link3270 server adapters only.

For Link3270 server adapters, the application data portion of the request message must be the Application Data Structure (ADS) of the target application. If the developer of the service application wants to use XML for the passthrough request message, he or she must adhere to specific guidelines that map the ADS to an XML schema. The following sample shows the XML passthrough schema of the request and response elements that follow the CICS Service Flow Runtime passthrough header.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.DFHMAXMI.com/schemas/DFHMAXMIInterface"
xmlns="http://www.w3.org/2001/XMLSchema" xmlns:scr="http://www.DFHMAXMI.com/schemas/DFHMAXMIInterface">
 <annotation>
     <documentation>
           ************************************************************************
         The format of each request/response begins with a main session header
         (DFHMAH).
         For a detailed description of the DFHMAH header,
         see "DFHMAH header structure" on page 80

         For a passthrough request/response a passthrough header (DFHMAH2)
         immediately follows the header (DFHMAH).

         For a detailed description of the DFHMAH2 header,
         see "DFHMAH2 header structure" on page 87

        This XML passthrough schema defines request/response elements that
         follow the passthrough header (DFHMAH2).
        </documentation>
        <documentation>
         Each XML passthrough requst/response contains a screen header
         (CIA-SCREEN-HEADER) that immediately follows DFHMAH2.

         For a detailed description of CIA-SCREEN-HEADER,
         see "CIA-SCREEN-HEADER header structure" on page 92

         The screen header (CIA-SCREEN-HEADER) contains the attentionID and the count of
         the number of maps, input fields, and text fields contained in the screen.
         Input fields and text fields do not correspond to the fields contained
         within a map.
        </documentation>
        <documentation>
         Request elements (screen, map, input):
        </documentation>
        <documentation>
         An unformatted or unmapped request is used to begin a new session.
         In this case, the "screen" element may contain a single "input" element.
         The value of the "input" element is the input text, beginning with the 4
         character transaction ID. Irregardless of whether an input element is
         present the target 3270 application will be run using the value contained
         in the passthrough request header "dfhmah2_transid" element.
        </documentation>
```

```
    <documentation>
     A mapped request is used to continue an existing session.
     In this case the "screen" element will contain a single "map" element.
    </documentation>
    <documentation>
     A "text" element is not valid for either a mapped or unmapped input
     request.
    </documentation>
    <documentation>
     Response elements (screen, map, htext, text, ttext):
    </documentation>
    <documentation>
     For a response the "screen" element may contain both "text" or "map"
     elements in any combination.
     A "map" element contains 1 or more "field" elements that correspond to
     named fields in a BMS map.
    </documentation>
    <documentation>
     The value of a "text" element contains text from a SEND TEXT command or
     from a SEND command that contains a single line of text.
    </documentation>
    <documentation>
     The value of a "htext" element contains header text from a SEND TEXT command.
    </documentation>
    <documentation>
     The value of a "ttext" element contains trailer text from a SEND TEXT command.
    </documentation>
    <documentation>
     An "input" element is not valid in a response.
     ********************************************************************************
    </documentation>
</annotation>
<element name="screen">
    <complexType>
     <sequence>
            <element maxOccurs="unbounded" minOccurs="0" ref="scr:map"/>
            <choice>
                <element maxOccurs="unbounded" minOccurs="0" ref="scr:text"/>
                <element maxOccurs="unbounded" minOccurs="0" ref="scr:htext"/>
                <element maxOccurs="unbounded" minOccurs="0" ref="scr:ttext"/>
                <element maxOccurs="1" minOccurs="0" ref="scr:input"/>
            </choice>
        </sequence>
        <attribute name="attentionid" use="required">
         <annotation>
          <documentation>
           attentionid= key pressed to submit the input screen
          </documentation>
         </annotation>
         <simpleType>
          <restriction base="string">
                 <enumeration value="enter"/>
                    <enumeration value="clear"/>
                    <enumeration value="pf1"/>
                    <enumeration value="pf2"/>
                    <enumeration value="pf3"/>
                    <enumeration value="pf4"/>
                    <enumeration value="pf5"/>
                    <enumeration value="pf6"/>
                    <enumeration value="pf7"/>
                    <enumeration value="pf8"/>
                    <enumeration value="pf9"/>
                    <enumeration value="pf10"/>
                    <enumeration value="pf11"/>
                    <enumeration value="pf12"/>
                    <enumeration value="pf13"/>
                    <enumeration value="pf14"/>
```

```
                    <enumeration value="pf15"/>
                    <enumeration value="pf16"/>
                    <enumeration value="pf17"/>
                    <enumeration value="pf18"/>
                    <enumeration value="pf19"/>
                    <enumeration value="pf20"/>
                    <enumeration value="pf21"/>
                    <enumeration value="pf22"/>
                    <enumeration value="pf23"/>
                    <enumeration value="pf24"/>
                    <enumeration value="pa1"/>
                    <enumeration value="pa2"/>
                    <enumeration value="pa3"/>
                    <enumeration value="pen"/>
                    <enumeration value="msre"/>
                    <enumeration value="opid"/>
                    <enumeration value="trig"/>
                    <maxLength value="5"/>
                </restriction>
               </simpleType>
            </attribute>
            <attribute name="maps" use="required">
             <annotation>
              <documentation>
               maps= number of maps, including input and text fields
              </documentation>
             </annotation>
             <simpleType>
                   <restriction base="integer">
                       <minInclusive value="0"/>
                       <totalDigits value="4"/>
                   </restriction>
               </simpleType>
            </attribute>
        </complexType>
    </element>
    <element name="map">
        <complexType>
            <sequence>
                <element maxOccurs="unbounded" minOccurs="1" ref="scr:field"/>
            </sequence>
            <attribute name="name" use="required">
             <annotation>
              <documentation>
               name= BMS map name
              </documentation>
             </annotation>
             <simpleType>
                   <restriction base="string">
                       <maxLength value="7"/>
                   </restriction>
               </simpleType>
            </attribute>
            <attribute name="mapset" use="required">
             <annotation>
              <documentation>
               mapset= BMS mapset name
              </documentation>
             </annotation>
             <simpleType>
                   <restriction base="string">
                       <maxLength value="7"/>
                   </restriction>
               </simpleType>
            </attribute>
            <attribute name="rows" use="optional">
             <annotation>
```

```
          <documentation>
           rows= number of map rows
          </documentation>
         </annotation>
         <simpleType>
               <restriction base="integer">
                   <minInclusive value="12"/>
                   <maxInclusive value="43"/>
                   <totalDigits value="2"/>
               </restriction>
            </simpleType>
        </attribute>
        <attribute name="columns" use="optional">
         <annotation>
          <documentation>
           columns= number of BMS map columns
          </documentation>
         </annotation>
 <simpleType>
               <restriction base="integer">
                   <minInclusive value="40"/>
                   <maxInclusive value="132"/>
                   <totalDigits value="3"/>
               </restriction>
            </simpleType>
        </attribute>
        <attribute name="fields" use="optional">
         <annotation>
          <documentation>
           fields= number of map fields
          </documentation>
         </annotation>
         <simpleType>
               <restriction base="integer">
                   <minInclusive value="1"/>
                   <totalDigits value="3"/>
               </restriction>
            </simpleType>
        </attribute>
        <attribute name="focus" use="optional">
         <annotation>
          <documentation>
           focus= map field name where the cursor
           is initially positioned
          </documentation>
         </annotation>
         <simpleType>
               <restriction base="string">
                   <maxLength value="32"/>
               </restriction>
            </simpleType>
        </attribute>
        <attribute name="length" use="optional">
         <annotation>
          <documentation>
           length= length of map ADS or text
          </documentation>
         </annotation>
         <simpleType>
               <restriction base="integer">
                   <minInclusive value="0"/>
                   <totalDigits value="4"/>
               </restriction>
            </simpleType>
        </attribute>
    </complexType>
</element>
```

```
<element name="field">
    <complexType>
     <annotation>
          <documentation>
           field= BMS map field
          </documentation>
         </annotation>
        <simpleContent>
            <extension base="string">
                <attribute name="name" use="required">
                 <simpleType>
            <annotation>
             <documentation>
              name= BMS field name
             </documentation>
            </annotation>
                  <restriction base="string">
                      <maxLength value="30"/>
                  </restriction>
              </simpleType>
          </attribute>
                <attribute name="ml" use="required">
                 <simpleType>
                  <annotation>
             <documentation>
              ml= maximum length
             </documentation>
            </annotation>
                  <restriction base="integer">
                      <minInclusive value="1"/>
                      <maxInclusive value="255"/>
                      <totalDigits value="3"/>
                  </restriction>
              </simpleType>
  </attribute>
                <attribute name="pa" use="required">
                    <simpleType>
                     <annotation>
             <documentation>
              pa= protect attribute:
              p= protect,
              u= unprotect
             </documentation>
            </annotation>
                        <restriction base="string">
                            <enumeration value="p"/>
                            <enumeration value="u"/>
                            <maxLength value="1"/>
                        </restriction>
                    </simpleType>
                </attribute>
                <attribute name="ia" use="required">
                    <simpleType>
                     <annotation>
             <documentation>
              ia= intensity attribute:
              n= normal intensity,
              b= bright intensity,
              d= dark or non-display intensity
             </documentation>
            </annotation>
                        <restriction base="string">
                            <enumeration value="n"/>
                            <enumeration value="b"/>
                            <enumeration value="d"/>
                            <maxLength value="1"/>
                        </restriction>
```

```
        </simpleType>
    </attribute>
    <attribute name="fa" use="optional">
        <simpleType>
         <annotation>
 <documentation>
  fa= FSET attribute:
  y= FSET on or modified data tag turned on
 </documentation>
</annotation>
            <restriction base="string">
                <enumeration value="y"/>
                <maxLength value="1"/>
            </restriction>
        </simpleType>
    </attribute>
    <attribute name="da" use="optional">
        <simpleType>
         <annotation>
 <documentation>
  da= selector pen detectable attribute:
  y= detectable turned on
 </documentation>
</annotation>
            <restriction base="string">
                <enumeration value="y"/>
                <maxLength value="1"/>
            </restriction>
        </simpleType>
    </attribute>
    <attribute name="ca" use="optional">
        <simpleType>
         <annotation>
 <documentation>
  ca= color attribute:
  b= blue,
  r= red,
  g= green,
  t= turqoise,
  y= yellow,
  p= pink,
  n= neutral
 </documentation>
</annotation>
            <restriction base="string">
                <enumeration value="b"/>
                <enumeration value="r"/>
                <enumeration value="g"/>
                <enumeration value="t"/>
                <enumeration value="y"/>
                <enumeration value="p"/>
                <enumeration value="n"/>
                <maxLength value="1"/>
            </restriction>
        </simpleType>
    </attribute>
    <attribute name="ha" use="optional">
        <simpleType>
         <annotation>
 <documentation>
  ha= highlight attribute:
  b= blink,
  u= underline,
  r= reverse video
 </documentation>
</annotation>
            <restriction base="string">
```

```
                    <enumeration value="b"/>
                    <enumeration value="u"/>
                    <enumeration value="r"/>
                    <maxLength value="1"/>
                </restriction>
            </simpleType>
        </attribute>
        <attribute name="va" use="optional">
            <simpleType>
             <annotation>
   <documentation>
    va= Validation attribute:
    f= must fill,
    e= must enter,
    t= trigger
   </documentation>
  </annotation>
                <restriction base="string">
                    <enumeration value="f"/>
                    <enumeration value="e"/>
                    <enumeration value="t"/>
                    <maxLength value="1"/>
                </restriction>
            </simpleType>
        </attribute>
        <attribute name="ps" use="optional">
         <simpleType>
          <annotation>
   <documentation>
    ps= programmed symbols
   </documentation>
  </annotation>
  <restriction base="string">
                    <maxLength value="1"/>
                </restriction>
 </simpleType>
</attribute>
      <attribute name="so" use="optional">
          <simpleType>
           <annotation>
   <documentation>
    so= shift in / shift out:
    y= yes,
    n= no
   </documentation>
  </annotation>
                <restriction base="string">
                    <enumeration value="y"/>
                    <enumeration value="n"/>
                    <maxLength value="1"/>
                </restriction>
            </simpleType>
        </attribute>
        <attribute name="oi" use="optional">
         <simpleType>
          <annotation>
   <documentation>
    oi= occurs index
   </documentation>
  </annotation>
        <restriction base="integer">
           <minInclusive value="1"/>
           <totalDigits value="3"/>
       </restriction>
     </simpleType>
 </attribute>
  </extension>
```

```
            </simpleContent>
        </complexType>
    </element>
    <element name="text">
     <simpleType>
      <annotation>
             <documentation>
              text= text from SEND TEXT or SEND command
             </documentation>
            </annotation>
            <restriction base="string">
                <maxLength value="4096"/>
            </restriction>
        </simpleType>
    </element>
    <element name="htext">
     <simpleType>
      <annotation>
             <documentation>
              htext= header text from SEND TEXT command
             </documentation>
            </annotation>
            <restriction base="string">
                <maxLength value="4096"/>
            </restriction>
        </simpleType>
    </element>
    <element name="ttext">
     <simpleType>
      <annotation>
             <documentation>
              ttext= trailer text from SEND TEXT command
             </documentation>
            </annotation>
            <restriction base="string">
                <maxLength value="4096"/>
            </restriction>
        </simpleType>
    </element>
    <element name="input">
     <simpleType>
      <annotation>
             <documentation>
              input= text entered on unformatted or unmapped screen
             </documentation>
            </annotation>
            <restriction base="string">
                <maxLength value="132"/>
            </restriction>
        </simpleType>
    </element>
</schema>
```

XML passthrough request and reply

The following sample demonstrates processing of the XML request sent into the installation verification program.

The sample shows two XML request and reply pairs, as follows:

- **Inbound request 1**"Inbound request 1" on page 356: A passthrough request in XML to launch the IVP application, (CMAV).
- **Outbound reply 1**"Outbound response to inbound request 1" on page 357: The resulting response, in XML, to the Inbound XML request 1. The response contains the Initial Customer Screen, (CIFS01).

- **Inbound request 2**“Inbound request 2” on page 358: A passthrough request in XML that inserts the search option **9** in Row22, Col 70 of the Customer Screen (CIFS01), along with the DFH-ENTER Key (attentionid = 'enter').
- **Outbound reply 2**“Response to inbound request 2” on page 359: The resulting reply in XML that contains the Customer Name Search Screen (CIFS09), 11 rows of data and a "More on next page" message.

Inbound request 1:

```
<?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>  </dfhmah__userid>
<dfhmah__format>MAH2</dfhmah__format>
<dfhmah__returncode>0</dfhmah__returncode>
<dfhmah__compcode>0</dfhmah__compcode>
<dfhmah__mode>0</dfhmah__mode>
<dfhmah__suspstatus>0</dfhmah__suspstatus>
<dfhmah__abendcode>  </dfhmah__abendcode>
<dfhmah__message>  </dfhmah__message>
<dfhmah__uowcontrol>3</dfhmah__uowcontrol>
<dfhmah__processtype>DFHMAINA</dfhmah__processtype>
<dfhmah__processname>  </dfhmah__processname>
<dfhmah__requestname>PASSTHRU</dfhmah__requestname>
<dfhmah__datalength>298</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>1</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>
<dfhmah2>
<dfhmah2__strucid>MAH2</dfhmah2__strucid>
<dfhmah2__version>1</dfhmah2__version>
<dfhmah2__struclength>164</dfhmah2__struclength>
<dfhmah2__reserved>  </dfhmah2__reserved>
<dfhmah2__format>  </dfhmah2__format>
<dfhmah2__datalength>134</dfhmah2__datalength>
<dfhmah2__transid>CMAV</dfhmah2__transid>
<dfhmah2__receive__type>0</dfhmah2__receive__type>
<dfhmah2__next__transid>  </dfhmah2__next__transid>
<dfhmah2__use__fkeeptime__ind>1</dfhmah2__use__fkeeptime__ind>
<dfhmah2__facilitykeeptime>3600</dfhmah2__facilitykeeptime>
<dfhmah2__facilitylike>  </dfhmah2__facilitylike>
<dfhmah2__getwaitinterval>300</dfhmah2__getwaitinterval>
<dfhmah2__vector__logging>0</dfhmah2__vector__logging>
<dfhmah2__deallocate__ind>3</dfhmah2__deallocate__ind>
<dfhmah2__send__aid__first>1</dfhmah2__send__aid__first>
<dfhmah2__initial__aid__byte>'</dfhmah2__initial__aid__byte>
<dfhmah2__clientip__address>  </dfhmah2__clientip__address>
<dfhmah2__resptime>0</dfhmah2__resptime>
<dfhmah2__applresptime>0</dfhmah2__applresptime>
<dfhmah2__xml__programid>  </dfhmah2__xml__programid>
<dfhmah2__reserved2>  </dfhmah2__reserved2>
```

```
</dfhmah2>
<dfhmaad>
<screen>
<input>CMAV</input>
</screen>
</dfhmaad></cbl:dfhmamsg>
```

Outbound response to inbound request 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<cbl:dfhmamsg xmlns:cbl="http://www.DFHMAXMO.com/schemas/DFHMAXMOInterface">
<dfhmah>
<dfhmah__strucid>MAH</dfhmah__strucid>
<dfhmah__version>2</dfhmah__version>
<dfhmah__struclength>384</dfhmah__struclength>
<dfhmah__userid> </dfhmah__userid>
<dfhmah__format>MAH2</dfhmah__format>
<dfhmah__returncode>0</dfhmah__returncode>
<dfhmah__compcode>1016</dfhmah__compcode>
<dfhmah__mode>1026</dfhmah__mode>
<dfhmah__suspstatus>1027</dfhmah__suspstatus>
<dfhmah__abendcode> </dfhmah__abendcode>
<dfhmah__message> </dfhmah__message>
<dfhmah__uowcontrol>3</dfhmah__uowcontrol>
<dfhmah__processtype>DFHMAINA</dfhmah__processtype>
<dfhmah__processname>SYSMKW10000652003311858327770</dfhmah__processname>
<dfhmah__requestname>PASSTHRU</dfhmah__requestname>
<dfhmah__datalength>1520</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>1</dfhmah__linktype>
<dfhmah__more__data__ind>0</dfhmah__more__data__ind>
<dfhmah__bridge__rc>0</dfhmah__bridge__rc>
<dfhmah__statetoken>0078000400000009</dfhmah__statetoken>
<dfhmah__reserved2> </dfhmah__reserved2>
</dfhmah>
<dfhmah2>
<dfhmah2__strucid>MAH2</dfhmah2__strucid>
<dfhmah2__version>1</dfhmah2__version>
<dfhmah2__struclength>164</dfhmah2__struclength>
<dfhmah2__reserved> </dfhmah2__reserved>
<dfhmah2__format>REPLY</dfhmah2__format>
<dfhmah2__datalength>2540</dfhmah2__datalength>
<dfhmah2__transid>CMAV</dfhmah2__transid>
<dfhmah2__receive__type>0</dfhmah2__receive__type>
<dfhmah2__next__transid> </dfhmah2__next__transid>
<dfhmah2__use__fkeeptime__ind>1</dfhmah2__use__fkeeptime__ind>
<dfhmah2__facilitykeeptime>3600</dfhmah2__facilitykeeptime>
<dfhmah2__facilitylike> </dfhmah2__facilitylike>
<dfhmah2__getwaitinterval>300</dfhmah2__getwaitinterval>
<dfhmah2__vector__logging>0</dfhmah2__vector__logging>
<dfhmah2__deallocate__ind>3</dfhmah2__deallocate__ind>
<dfhmah2__send__aid__first>1</dfhmah2__send__aid__first>
<dfhmah2__initial__aid__byte>&amp;apos;</dfhmah2__initial__aid__byte>
<dfhmah2__clientip__address> </dfhmah2__clientip__address>
<dfhmah2__resptime>850</dfhmah2__resptime>
<dfhmah2__applresptime>0</dfhmah2__applresptime>
<dfhmah2__xml__programid>
<dfhmah2__xml__program> </dfhmah2__xml__program>
<dfhmah2__xml__program__tag> </dfhmah2__xml__program__tag>
```

```
</dfhmah2__xml__programid>
<dfhmah2__reserved2> </dfhmah2__reserved2>
</dfhmah2>
<dfhmaad>
<screen attentionid="enter" maps="001">
<map name="MAPA" mapset="DFHMAB1" rows="24" columns="80" fields="42" focus="AR03C18">
<field name="AR01C25" ml="31" pa="p" ia="n">*** CUSTOMER IDENTIFICATION ***</field>
<field name="AR01C70" ml="10" pa="p" ia="n">CIFS01</field>
<field name="AR02C68" ml="10" pa="p" ia="n"/>
<field name="AR03C18" ml="5" pa="u" ia="b" da="y" ha="u"/>
<field name="AR04C18" ml="30" pa="p" ia="n" ha="u"/>
<field name="AR05C18" ml="35" pa="p" ia="n" ha="u"/>
<field name="AR06C18" ml="20" pa="p" ia="n" ha="u"/>
<field name="AR06C48" ml="2" pa="p" ia="n" ha="u"/>
<field name="AR06C63" ml="5" pa="p" ia="n" ha="u">00000</field>
<field name="AR07C20" ml="3" pa="p" ia="n" ha="u">000</field>
<field name="AR07C27" ml="3" pa="p" ia="n" ha="u">000</field>
<field name="AR07C33" ml="4" pa="p" ia="n" ha="u">0000</field>
<field name="AR08C18" ml="1" pa="p" ia="n" ha="u"/>
<field name="AR08C37" ml="30" pa="p" ia="n" ha="u"/>
<field name="AR10C18" ml="30" pa="p" ia="n" ha="u"/>
<field name="AR11C18" ml="30" pa="p" ia="n" ha="u"/>
<field name="AR12C18" ml="20" pa="p" ia="n" ha="u"/>
<field name="AR12C48" ml="2" pa="p" ia="n" ha="u"/>
<field name="AR12C63" ml="5" pa="p" ia="n" ha="u">00000</field>
<field name="AR13C20" ml="3" pa="p" ia="n" ha="u">000</field>
<field name="AR13C27" ml="3" pa="p" ia="n" ha="u">000</field>
<field name="AR13C33" ml="4" pa="p" ia="n" ha="u">0000</field>
<field name="AR16C12" ml="1" pa="p" ia="d" ha="u"/>
<field name="AR16C20" ml="15" pa="p" ia="n" ha="u"/>
<field name="AR16C49" ml="7" pa="p" ia="n" ha="u"/>
<field name="AR17C12" ml="1" pa="p" ia="d" ha="u"/>
<field name="AR17C20" ml="15" pa="p" ia="n" ha="u"/>
<field name="AR17C49" ml="7" pa="p" ia="n" ha="u"/>
<field name="AR18C12" ml="1" pa="p" ia="d" ha="u"/>
<field name="AR18C20" ml="15" pa="p" ia="n" ha="u"/>
<field name="AR18C49" ml="7" pa="p" ia="n" ha="u"/>
<field name="AR19C12" ml="1" pa="p" ia="d" ha="u"/>
<field name="AR19C20" ml="15" pa="p" ia="n" ha="u"/>
<field name="AR19C49" ml="7" pa="p" ia="n" ha="u"/>
<field name="AR20C12" ml="1" pa="p" ia="d" ha="u"/>
<field name="AR20C20" ml="15" pa="p" ia="n" ha="u"/>
<field name="AR20C49" ml="7" pa="p" ia="n" ha="u"/>
<field name="AR22C02" ml="65" pa="p" ia="n">Select an option (1=Display  2=Add  3=Alter  4=Delete  9=Search)</field>
<field name="AR22C70" ml="1" pa="u" ia="b" fa="y" da="y" ha="u"/>
<field name="AR23C02" ml="75" pa="p" ia="n">Enter=Process F3=Exit F12=Cancel</field>
<field name="AR24C02" ml="75" pa="p" ia="n"/>
<field name="AR24C79" ml="1" pa="p" ia="d" fa="y"/>
</map>
</screen>
</dfhmaad>
</cbl:dfhmamsg>
```

Inbound request 2:

```
<?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>  </dfhmah__userid>
<dfhmah__format>MAH2</dfhmah__format>
<dfhmah__returncode>0</dfhmah__returncode>
<dfhmah__compcode>0</dfhmah__compcode>
<dfhmah__mode>0</dfhmah__mode>
<dfhmah__suspstatus>0</dfhmah__suspstatus>
<dfhmah__abendcode>  </dfhmah__abendcode>
```

```
<dfhmah__message>  </dfhmah__message>
<dfhmah__uowcontrol>3</dfhmah__uowcontrol>
<dfhmah__processtype>DFHMAINA</dfhmah__processtype>
<dfhmah__processname>  </dfhmah__processname>
<dfhmah__requestname>PASSTHRU</dfhmah__requestname>
<dfhmah__datalength>1184</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>1</dfhmah__linktype>
<dfhmah__more__data__ind>0</dfhmah__more__data__ind>
<dfhmah__bridge__rc>0</dfhmah__bridge__rc>
<dfhmah__statetoken>0078000400000009</dfhmah__statetoken>
<dfhmah__reserved2>  </dfhmah__reserved2>
</dfhmah>
<dfhmah2>
<dfhmah2__strucid>MAH2</dfhmah2__strucid>
<dfhmah2__version>1</dfhmah2__version>
<dfhmah2__struclength>164</dfhmah2__struclength>
<dfhmah2__reserved>   </dfhmah2__reserved>
<dfhmah2__format>  </dfhmah2__format>
<dfhmah2__datalength>1020</dfhmah2__datalength>
<dfhmah2__transid>CMAV</dfhmah2__transid>
<dfhmah2__receive__type>1</dfhmah2__receive__type>
<dfhmah2__next__transid>  </dfhmah2__next__transid>
<dfhmah2__use__fkeeptime__ind>1</dfhmah2__use__fkeeptime__ind>
<dfhmah2__facilitykeeptime>3600</dfhmah2__facilitykeeptime>
<dfhmah2__facilitylike>  </dfhmah2__facilitylike>
<dfhmah2__getwaitinterval>300</dfhmah2__getwaitinterval>
<dfhmah2__vector__logging>0</dfhmah2__vector__logging>
<dfhmah2__deallocate__ind>3</dfhmah2__deallocate__ind>
<dfhmah2__send__aid__first>0</dfhmah2__send__aid__first>
<dfhmah2__initial__aid__byte> </dfhmah2__initial__aid__byte>
<dfhmah2__clientip__address>  </dfhmah2__clientip__address>
<dfhmah2__resptime>0</dfhmah2__resptime>
<dfhmah2__applresptime>0</dfhmah2__applresptime>
<dfhmah2__xml__programid>  </dfhmah2__xml__programid>
<dfhmah2__reserved2>  </dfhmah2__reserved2>
</dfhmah2>
<dfhmaad>
<screen attentionid="enter" maps="1">
<map name="MAPA" mapset="DFHMAB1" rows="24" columns="80" fields="1">
<field name="AR22C70">9</field>
</map>
</screen>
</dfhmaad>
</cbl:dfhmamsg>
```

Response to inbound request 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<cbl:dfhmamsg xmlns:cbl="http://www.DFHMAXMO.com/schemas/DFHMAXMOInterface">
<dfhmah>
<dfhmah__strucid>MAH</dfhmah__strucid>
<dfhmah__version>2</dfhmah__version>
<dfhmah__struclength>384</dfhmah__struclength>
<dfhmah__userid> </dfhmah__userid>
<dfhmah__format>MAH2</dfhmah__format>
<dfhmah__returncode>0</dfhmah__returncode>
<dfhmah__compcode>1016</dfhmah__compcode>
<dfhmah__mode>1026</dfhmah__mode>
```

```
<dfhmah__suspstatus>1027</dfhmah__suspstatus>
<dfhmah__abendcode> </dfhmah__abendcode>
<dfhmah__message> </dfhmah__message>
<dfhmah__uowcontrol>3</dfhmah__uowcontrol>
<dfhmah__processtype>DFHMAINA</dfhmah__processtype>
<dfhmah__processname>SYSMKW10000656003311858542820</dfhmah__processname>
<dfhmah__requestname>PASSTHRU</dfhmah__requestname>
<dfhmah__datalength>349</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>1</dfhmah__linktype>
<dfhmah__more__data__ind>0</dfhmah__more__data__ind>
<dfhmah__bridge__rc>0</dfhmah__bridge__rc>
<dfhmah__statetoken>0078000400000009</dfhmah__statetoken>
<dfhmah__reserved2> </dfhmah__reserved2>
</dfhmah>
<dfhmah2>
<dfhmah2__strucid>MAH2</dfhmah2__strucid>
<dfhmah2__version>1</dfhmah2__version>
<dfhmah2__struclength>164</dfhmah2__struclength>
<dfhmah2__reserved> </dfhmah2__reserved>
<dfhmah2__format>REPLY</dfhmah2__format>
<dfhmah2__datalength>1593</dfhmah2__datalength>
<dfhmah2__transid>CMAV</dfhmah2__transid>
<dfhmah2__receive__type>1</dfhmah2__receive__type>
<dfhmah2__next__transid> </dfhmah2__next__transid>
<dfhmah2__use__fkeeptime__ind>1</dfhmah2__use__fkeeptime__ind>
<dfhmah2__facilitykeeptime>3600</dfhmah2__facilitykeeptime>
<dfhmah2__facilitylike> </dfhmah2__facilitylike>
<dfhmah2__getwaitinterval>300</dfhmah2__getwaitinterval>
<dfhmah2__vector__logging>0</dfhmah2__vector__logging>
<dfhmah2__deallocate__ind>3</dfhmah2__deallocate__ind>
<dfhmah2__send__aid__first>0</dfhmah2__send__aid__first>
<dfhmah2__initial__aid__byte> </dfhmah2__initial__aid__byte>
<dfhmah2__clientip__address> </dfhmah2__clientip__address>
<dfhmah2__resptime>5650</dfhmah2__resptime>
<dfhmah2__applresptime>0</dfhmah2__applresptime>
<dfhmah2__xml__programid>
<dfhmah2__xml__program> </dfhmah2__xml__program>
<dfhmah2__xml__program__tag> </dfhmah2__xml__program__tag>
</dfhmah2__xml__programid>
<dfhmah2__reserved2> </dfhmah2__reserved2>
</dfhmah2>
<dfhmaad>
<screen attentionid="enter" maps="001">
<map name="CIAMPA" mapset="DFHMAB1" rows="24" columns="80" fields="24" focus="">
<field name="SR05C09" ml="15" pa="u" ia="n" fa="y" ha="u"/>
<field name="BR09C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR09C03" ml="77" pa="p" ia="n">ONE CUSTOMER</field>
<field name="BR10C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR10C03" ml="77" pa="p" ia="n">11111 11111</field>
<field name="BR11C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR11C03" ml="77" pa="p" ia="n">12341</field><field name="BR12C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR12C03" ml="77" pa="p" ia="n">2222222222222</field>
<field name="BR13C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR13C03" ml="77" pa="p" ia="n">23456</field>
<field name="BR14C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR14C03" ml="77" pa="p" ia="n">3333333333333</field>
<field name="BR15C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR15C03" ml="77" pa="p" ia="n">34567</field>
```

```
<field name="BR16C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR16C03" ml="77" pa="p" ia="n">4444444444444</field>
<field name="BR17C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR17C03" ml="77" pa="p" ia="n">45678</field>
<field name="BR18C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR18C03" ml="77" pa="p" ia="n">5555555555555</field>
<field name="BR19C01" ml="1" pa="u" ia="b" da="y" ha="r"/>
<field name="BR19C03" ml="77" pa="p" ia="n">56789</field>
<field name="CR24C01" ml="77" pa="p" ia="n" ha="r">MORE ON NEXT PAGE</field>
</map>
</screen>
</dfhmaad>
</cbl:dfhmamsg>
```

Part 3. Appendixes

© Copyright IBM Corp. 2001, 2010

Appendix. Migrating from MQSeries Integrator Agent for CICS Transaction Server (MQIAC)

You can run your existing MQIAC adapter services in CICS Service Flow Runtime without needing to regenerate or recompile them. However, it is recommended that you regenerate your Link3270 adapter services before deploying them in the runtime environment.

There are a number of migration points that you need to be aware of, and these are explained in the following section.

Deploying MQIAC adapter services

When you deploy an MQIAC server adapter in CICS SFR, you define it to the runtime environment by updating the Property file using DFHMAMPU.

If you are deploying Link3270 server adapters, you need to perform the additional deployment steps that are listed as optional in the sequence of steps below.

1. Edit DFHMAMPU, adding parameters **PARM04** through **PARM07** to the Type R request. For example:

```
//SYSIN DD *
..

TYPE=R
NAME=TCMQ01
PARM01=0
PARM02=TCMNAV1
PARM03=TCMN
PARM04=0
PARM05=
PARM06=2
PARM07=0
PARMXX
/*
//
```

2. If you are deploying a Link3270 server adapter and you require AOR routing, update the **PARM07** parameter of the Type 5 request in DFHMAMPU with a `Y` value.
3. Run DFHMAMPU using the update job generated by the MQIAC builder to update the Property file. The update job is called @*numeric value*.jcp.
4. Optional: If you are deploying Link3270 server adapters, you need to perform the following additional steps:
   a. Create the Link3270 repository file DFHMALRF using the job DFHMADDR with the MQIAC copybooks. Alternatively, you can edit DFHMADDL to point to MQIAC runtime libraries.
   b. Update DFHMALRF to include the necessary mapsets using the sample job DFHMAMLU.

Request processing of MQIAC adapter services

When you run MQIAC adapter services in the CICS Service Flow Runtime, the runtime environment supports the services differently to adapter services modeled in Service Flow Modeler.

© Copyright IBM Corp. 2001, 2010

The runtime environment supports the invocation of MQIAC adapter services by differentiating which services to run through the DFHMAH-VERSION field in the request message header. If the field has a value of 1, the runtime environment knows that a MQIAC server adapter is being invoked and processes the request accordingly.

MQIAC server adapters that use the Link3270 bridge

It is recommended that you recompile and redeploy MQIAC server adapters that use the Link3270 bridge.

In flows generated by Service Flow Modeler, transaction routing is automatically handled. However in Link3270 adapter request processing for MQIAC adapter services, you should ensure that the AOR routing indicator is set to YES in the MQIAC flows. This indicator configures the runtime environment to use transaction routing for Link3270 adapter services. The AOR routing indicator is a Link3270 node property called MAT_AOR_ROUTING.

Set the AOR routing indicator to NO for all adapter program properties files and .rsc files when all target CICS application transactions are run locally in the region for Link3270 processing.

Prior MQIAC generated Link3270 navigator file read errors, file rewrite errors, and file delete errors are reported by the generated Link3270 navigator program.

If you want to switch vector logging on, you can only set the value of 1. A value of 2 is not supported. Also, if you do not recompile and redeploy your MQIAC server adapters that use the Link3270 bridge, vector logging continues to use the DFHMALVF file, rather than using DFHMALVA and DFHMALVB. To view the DFHMALVF vector log, run the DFHMAMVM JCL. This runs the program DFHMAVUM to dump the vector log file.

Processing non-unique user ids

Processing of non-unique user ids is the same for FEPI and Link3270 adapters. Using unique user ids in MQIAC enhances the processing of Link3270 adapter processing only, as the business state management is performed differently to CICS Service Flow Runtime. With unique user ids MQIAC users can use the Link3270 bridge facilities and associated session data in one request, leave it assigned to the unique user id and reuse it in subsequent request processing. CICS Service Flow Runtime provides this benefit without requiring the unique user id.

Link3270 bridge facility assignment processing for non-unique user ids

A non-unique user id is one that is not associated to a single owner, but instead can be used by more than one user. This function is support for compatibility in the runtime environment.

Link3270 State file, DFHMALSF, assignment for non-unique user ids provides the following processing:

- Multiple Link3270 bridge facilities can be assigned uniquely to a non-unique user id. These facilities can be left assigned to that user id during invocation of potentially multiple Link3270 server adapters within the scope of a single executing business request.

- That multiple of these business requests may be executing concurrently for the same non-unique user id. The processing of each of these concurrent business requests must be performed in separate and distinct BTS processes.
- The provision for Link3270 State file records remain assigned, (for a specific **unique user id**) across multiple invocations of subsequent processes that may require usage of that Link3270 bridge facility. This means the CICS Service Flow Runtime Link3270 State file could remain assigned to the user id after the business request process is complete.

Link3270 bridge facilities are required to run CICS applications in modeled flows that contain Link3270 server adapters. Link3270 bridge facility and state file processing for non-unique user id looks at processing requirements that arise when non-unique user ids are used with CICS Service Flow Runtime.

Using non-unique user ids in your Link3270 service flows will determine how CICS Service Flow Runtime assigns Link3270 State file information for allocated Link3270 bridge facilities that are used to execute CICS applications via the CICS Link3270 bridge mechanism. See“Run time processing of non-unique user IDs using Link3270 bridge server adapters” for a description of run time processing associated with LU assignments for non-unique user ids.

Implementing facility assignment processing for non-unique UserIDs with Link3270 bridge server adapters

At build time, the person who models the adapter can enable non-unique UserIDs for a Link3270 service flow. For information on how to enable processing for non-unique UserIDs, see the section on using property files of the Service Flow Modeler help in the *WebSphere Developer for System z* information center.

When you enable processing for non-unique UserIDs in your adapter model, you must also provide instructions in your adapter for the deletion of Link3270 bridge facilities and associated Link3270 session state data. All allocated facilities that were used by all of the Link3270 server adapters, and executed as part of a process, must be deleted. If you do not delete the bridge facilities, CICS Service Flow Runtime Link3270 State file records may remain assigned and will be rendered useless to standard CICS Service Flow Runtime processing. This is true regardless of the processing mode; synchronous processing for inquiry requests; or asynchronous processing for update requests. For information on how to set the Link3270 bridge facility deallocation indicator, see information on **MAT_DEACT_ON_EXIT** in the Service Flow Modeler component of the *WebSphere Developer for System z* information center.

Run time processing of non-unique user IDs using Link3270 bridge server adapters

Link3270 bridge server adapter processing for non-unique user IDs assumes that the user IDs at your site are not unique. When you use non-unique user IDs, you will not be able to take advantage of some CICS Service Flow Runtime processing capabilities.

Note: See “Why you should use unique user ids with CICS Service Flow Runtime” on page 178 for information about the benefits of unique UserIDs.

In a configuration where the user UserID is not unique, CICS Service Flow Runtime processing ensures that Link3270 bridge facility session state data is *stored uniquely* to the non-unique UserID. CICS Service Flow Runtime accomplishes this

by appending a unique tag to the Link3270 State file (DFHMALSF) owner assignment. This unique tag is comprised of the CICS Applid and the EIBTASKN of the DPL Stub program (DFHMADPL).

The unique tag, which is retrieved in the DPL Stub program (DFHMADPL), is used in conjunction with the user ID as part of the Link3270 State file assignment during process execution. The runtime processing of a non-unique user ID using Link3270 bridge server adapters is as follows:

- In the first Link3270 server adapter invoked during the execution of a process, the owner assignment is updated on the Link3270 State file (DFHMALSF) key using the unique tag in conjunction with the user ID.
- Any subsequent Link3270 server adapters executed as part of the same process will use this same unique tag identifier to retrieve the correct Link3270 bridge facility and associated session state data.
- The assignment indicator configuration remains in effect for the duration of the business request, which may include the invocation of multiple Link3270 server adapters.
- The unique tag identifier will allow for concurrent process execution by the same non-unique user ID, and it will not change during the execution of a single process.

If the process execution terminates abnormally, the service requestor is responsible for handling the condition by one of the following methods:

- initiating a process of a modeled flow to *compensate* for the failed process
- issuing a *cancel* request to the CICS Service Flow Runtime to release BTS resources.

See “Deciding on whether to cancel or run compensating flow” on page 60 for more information.

In addition to the processing associated with Link3270 bridge facility assignment for non-unique user IDs, CICS Service Flow Runtime provides *cleanup* logic in the DPL Stub program and in the Navigation Manager (DFHMAMGR). This cleanup logic is invoked when the process terminates abnormally resulting in allocated Link3270 bridge facilities and associated state data. See “Link3270 State file processing for non-unique IDs — synchronous mode” on page 369 and “Link3270 State file processing for non-unique IDs — asynchronous mode” on page 371 for a description of how the DPL Stub program implements cleanup logic to remove the bridge facility state data that remains on the CICS Service Flow Runtime Link3270 State file (DFHMALSF) after a process terminates abnormally.

Link3270 State file processing for unique IDs — synchronous mode

When you use unique UserIDs the Link3270 bridge facilities (and associated state data) are allocated and deleted as modeled in the Service Flow Modeler. The CICS Service Flow Runtime server run-time makes no attempts to locate and cleanup Link3270 State file records left assigned upon successful or unsuccessful execution of request processes.

Since only the UserID is used and is assumed unique in this mode, any and all Link3270 bridge facilities (and associated state data) allocated for the specific UserID can be located and used in the execution of subsequent request processes by that UserID.

Note: If Link3270 bridge facilities are left allocated in this mode for use in subsequent process execution, the service requestor is responsible to delete Link3270 bridge facilities for a specific UserID when use of that UserID is no longer required.

For example, as a normal end of day processing strategy, the service requestor could invoke a modeled flow as a process to locate and delete any allocated bridge facilities. Failure to incorporate a cleanup strategy for a specific user may not leave bridge facilities available in high volume environments over time to other users — unless the customer configuration is such that the bridge facility *maximum keeptime*, (BRIH-FACILITYKEEPTIME), is set such that bridge facilities are deleted in a timely fashion for that customer environment. For information on how to set the Link3270 bridge facility *maximum keeptime*, see information on the **MAT_MAX_FAC_KEEPTIME** property in the Service Flow Modeler help. For a definition of (BRIH-FACILITYKEEPTIME), see the *CICS External Interfaces Guide Version 2 Release 2* .

Link3270 State file processing for non-unique IDs — synchronous mode

If your site uses non-unique UserIDs, and if the process terminates abnormally due to a CICS Service Flow Runtime system error and / or an incorrectly modeled flow, this can result in allocated Link3270 bridge facilities and associated Link3270 State file data on CICS Service Flow Runtime Link3270 State file (DFHMALSF).

The CICS Service Flow Runtime / BTS process that was initiated to execute that modeled flow will, upon issuing a reply to the service requestor, *complete*.

Under the complete state there will be no CICS Service Flow Runtime / BTS process container information available for subsequent use to locate, use and / or delete bridge facilities and associated state data.

For this reason, the bridge facilities (and associated state data) will be deleted before the CICS Service Flow Runtime / BTS process is complete, successfully executed or not.

The CICS Service Flow Runtime DPL Stub program (DFHMADPL), will delete these facilities and associated state data before issuing the reply of an executed process to the controlling application.

The DPL Stub program uses the unique tag (see “Run time processing of non-unique user IDs using Link3270 bridge server adapters” on page 367), in conjunction with the user UserID to:

- locate all bridge facilities used in the process.
- delete all allocated bridge facilities.
- delete the associated bridge facility state data from the CICS Service Flow Runtime Link3270 State file record(s)

The CICS Service Flow Runtime / BTS process will then be completed.

Note: If the delete of bridge facilities fails, those bridge facilities will remain until the expiration of bridge facility *maximum keeptime*, (BRIH-FACILITYKEEPTIME). For information on how to set the Link3270 bridge facility *maximum keeptime*, see information on **MAT_MAX_FAC_KEEPTIME** in the Service Flow Modeler help.

Link3270 State file processing for unique IDs — asynchronous mode

When you use unique UserIDs the Link3270 bridge facilities (and associated state data) are allocated and deleted as modeled in the Service Flow Modeler. The CICS Service Flow Runtime server run-time makes no attempts to locate and cleanup Link3270 State file records left assigned upon successful or unsuccessful execution of request processes.

It is still necessary to *compensate* or *cancel* CICS Service Flow Runtime / BTS processes as the result of unsuccessful process execution. However, in this mode it is not necessary that the modeled flow include logic to delete allocated Link3270 bridge facilities (and associated Link3270 bridge facility state data).

Since only the UserID is used and is assumed to be unique, any and all allocated bridge facilities and associated bridge facility state data allocated to the specific UserID can be located and used in the execution of subsequent processes by that UserID.

Also if the only requirement, as determined by the service requestor, is to *cancel* an unsuccessfully executed process, the DPL Stub program (DFHMADPL) issues the BTS CANCEL ACQPROCESS command to end and complete the failed CICS Service Flow Runtime / BTS process.

The DPL Stub program (DFHMADPL) will not delete allocated bridge facilities or associated facility state data on the CICS Service Flow Runtime Link3270 State file because the UserID is unique and can be found in any subsequent business request/process execution issued by that same UserID.

The service requestor may want to *cancel* the failed CICS Service Flow Runtime BTS process but not delete bridge facilities or associated state data assigned to that UserID.

Subsequent modeled flows can be initiated as processes to locate, use and / or delete any allocated bridge facilities as desired.

Note: As was the case in synchronous processing for unique UserIDs, if bridge facilities are left allocated in this mode for use in subsequent process execution, the service requestor is responsible to delete Link3270 bridge facilities for a specific UserID when use of that UserID is no longer required.

For example, as a normal end of day processing strategy, the service requestor could invoke a modeled flow as a process to locate and delete any allocated bridge facilities. Failure to incorporate a cleanup strategy for a specific user may not leave bridge facilities available in high volume environments over time to other users — unless the customer configuration is such that the bridge facility *maximum keeptime*, (BRIH-FACILITYKEEPTIME), is set such that bridge facilities are deleted in a timely fashion for that customer environment. For information on how to set the Link3270 bridge facility *maximum keeptime*, see information on **MAT_MAX_FAC_KEEPTIME** in the Service Flow Modeler help. For a definition of (BRIH-FACILITYKEEPTIME), see *CICS External Interfaces Guide Version 2 Release 2*.

Link3270 State file processing for non-unique IDs — asynchronous mode

If your site uses non-unique UserIDs, and if the process terminates abnormally due to a CICS Service Flow Runtime system error or an incorrectly modeled flow, this can result in bridge facilities and associated facility state data remaining allocated.

An abnormal termination of a process does not complete/end the CICS Service Flow Runtime / BTS process that was initiated for that modeled flow. This leaves CICS Service Flow Runtime / BTS process container information available from the failed process for subsequent use. This is true regardless of whether you are using non-unique or unique UserIDs.

These failed CICS Service Flow Runtime / BTS processes must be addressed by the service requestor, again regardless of the type of UserID or regardless of whether the Link3270 bridge mechanism is implemented as part of the modeled flow. Since the CICS Service Flow Runtime / BTS process is not complete, BTS resources are still allocated with the state (business request state) of the CICS Service Flow Runtime / BTS process and process container information.

In order to complete the failed CICS Service Flow Runtime / BTS process and release those BTS resources, the service requestor must perform one of the following actions:

- initiate a process of a modeled compensating flow

  Note: The compensating flow must be run in the same mode as the original flow. This means if the original flow was configured to use non-unique UserID, you need to configure your compensating flow to use the LU assignment processing for non-unique UserIDs.
- issue a *cancel* request of the failed process to the CICS Service Flow Runtime server run-time to release the BTS resources and complete / end the CICS Service Flow Runtime process.

If the service requestor initiates a process of a modeled compensating flow, the DPL Stub program will:

- ACQUIRE the failed process using the failed process name and process type
- initiate a new CICS Service Flow Runtime / BTS process under which the compensating flow will run
- retrieve (issue GET CONTAINER) the information stored in the containers of the failed process
- issue a BTS CANCEL ACQPROCESS command thus completing the failed CICS Service Flow Runtime / BTS process and releasing BTS resources.
- place that information into the containers associated with the new CICS Service Flow Runtime / BTS process
- RUN the compensating flow process as modeled in the Service Flow Modeler

The unique tag used in conjunction with the non-unique UserID is included in the container information retrieved from the failed CICS Service Flow Runtime / BTS process and is available to the compensating process.

The unique tag is used to locate all allocated bridge facilities and associated facility state data used in the failed process.

The bridge facilities are used in the compensating process as indicated in the flow model. The facilities ultimately, however, must be deleted as part of that compensating process.

If the service requestor issues a *cancel* of the failed process, the DPL Stub program will:

- ACQUIRE the failed CICS Service Flow Runtime / BTS process thus gaining access to the container information of that failed CICS Service Flow Runtime / BTS process.

  The unique tag used in Link3270 State file processing for non-unique UserIDs is included in this container information.
- Use the unique tag retrieved in conjunction with the user UserID to locate all facilities used in the failed process.
- delete all facilities and associated facility state data on the CICS Service Flow Runtime Link3270 State file.
- issue a BTS CANCEL ACQPROCESS command thus completing the failed CICS Service Flow Runtime / BTS process and releasing BTS resources.

Note: If the delete of bridge facilities fails, those bridge facilities will remain until the expiration of bridge facility *maximum keeptime*, (BRIH-FACILITYKEEPTIME). For information on how to set the Link3270 bridge facility *maximum keeptime*, see information on **MAT_MAX_FAC_KEEPTIME** in the Service Flow Modeler help.

Note: Incorrectly modeled flows that leave facilities allocated in this mode of processing and that complete successfully, cause bridge facilities to remain allocated until the expiration of the bridge facility *maximum keeptime*, (BRIH-FACILITYKEEPTIME).

Cleanup logic in the CICS Service Flow Runtime Navigation Manager program (DFHMAMGR), similar to the logic added to the CICS Service Flow Runtime DPL Stub program (DFHMADPL), attempts to locate and delete facilities left allocated upon successful process execution. If, however, the delete of bridge facilities fails, those bridge facilities will remain until the expiration of bridge facility *maximum keeptime*, (BRIH-FACILITYKEEPTIME). For information on how to set the Link3270 bridge facility *maximum keeptime*, see information on **MAT_MAX_FAC_KEEPTIME** in the Service Flow Modeler help.

Error processing using WebSphere MQ

If you are using the WebSphere MQ system error queue configuration to handle error processing, you should be aware of the following points.

- The queues to which errors are written must be defined and enabled when you customize the CICS Service Flow Runtime. See the *MQSeries Integrator Agent for CICS User's Guide* for information on how to define and enable these queues.
- The Error listener program DFHMAERR maps the layout of the error file DFHMAERF using the record description copybook DFHMARER found in the **.SCIZMAC** product distribution library. CICS SFR comes with a dump utility for the error file. See “Troubleshooting checklist” on page 202 for information on how to run the utility and see “Error file dump” on page 283 for an annotated sample of the error file dump.
- If you are using the WebSphere MQ system error queue, the Error Listener program waits on the **CIA.SYSTEM.ERROR.QUEUE**. It does not wait on the

CIA.SYSTEM.DEAD.LETTER.QUEUE. The Error Listener wait interval for messages is specified on the process definition for the CIA.SYSTEM.ERROR.QUEUE. The wait interval is specified in seconds and must be the first 3 bytes of the User data. A value of 999 translates into MQWI-UNLIMITED.

- The default definition, CIA.SYSTEM.ERROR.PROCESS, specifies 60 seconds.
- If the GET fails, CompCode = MQCC-FAILED, for any reason other than MQRC-NO-MSG-AVAILABLE, a record is written to the error file DFHMAERF indicating the error and request processing is stopped.
- For condition MQRC-NO-MSG-AVAILABLE, processing is stopped without a record being written to the error file DFHMAERF.
- If the GET completes with a warning, CompCode = MQCC-WARNING, a record detailing the problem is written to the error file DFHMAERF. The retrieved message is placed on the CIA.SYSTEM.DEAD.LETTER.QUEUE, and processing is continues.

  The options, MQGMO-OPTIONS, used on the MQGET are:

  – MQGMO-WAIT
  – MQGMO-SYNCPOINT
  – MQGMO-FAIL-IF-QUIESCING
  – MQGMO-ACCEPT-TRUNCATED-MSG
- If the CIA.SYSTEM.ERROR.QUEUE does not exist, or if it has not been properly defined, the Error Logging Facility writes an error to the console indicating the queue error and tries to write the original error to the CIA.SYSTEM.DEAD.LETTER.QUEUE. If it writes to the CIA.SYSTEM.DEAD.LETTER.QUEUE successfully, error processing is complete. The error message remains on the CIA.SYSTEM.DEAD.LETTER.QUEUE and it is your choice regarding what action to take.
- When there is a problem reading or writing messages from or to the error queues, a message that indicates that a problem exists with the particular error queue, is written to the system console. Errors are written to the console with the EVENTUAL option to allow console scrolling. Errors will need to be deleted from the console by an authorized systems person.

  See “Conditions that cause CICS Service Flow Runtime to write to the system console” on page 197 for more information on when CICS SFR will write a message to the system console and for an example of the format of the messages that is written to the console.
- If you want to perform additional processing of error related information, you can develop your own error listener program to replace the one provided. For example, you could program an *Error Listener* to write error records to other environments (Tivoli, for example).
- If an error occurs when writing to the error file DFHMAERF, a message is written to the console detailing the problem, the error queue is set to MQQA-GET-INHIBITED and MQTC-ON (trigger control = on) and request processing is stopped. The record remains on the CIA.SYSTEM.ERROR.QUEUE. When the problem is fixed, the error queue should be set to Get enabled and any messages on the queue drained to the error file DFHMAERF.

Conditions for writing to the system console

Messages are written to the system console in the following conditions:

- A problem occurs writing to the transient data error queue CMAQ in the runtime environment during request processing.

- A problem occurs writing a run time processing error (a FILE write error) to the error file DFHMAERF in the Error Listener.
- An attempt to write an error message to the dead letter queue, CIA.SYSTEM.DEAD.LETTER.QUEUE, fails in the Error Listener (DFHMAERR; transaction ID = CMAE). The write to the dead letter queue is attempted after an MQGET call to the error queue, CIA.SYSTEM.ERROR.QUEUE, completes with a warning, MQCC-WARNING.
- A problem occurs writing to the error queues CIA.SYSTEM.ERROR.QUEUE or CIA.SYSTEM.DEAD.LETTER.QUEUE in the runtime during request processing.

See “Error processing” on page 195 for an explanation of how CICS SFR works to process error messages.

Format of MQGET error written to console

In this example, the following error:

```
CIA08004E MQGET error reported in CICS SFR transaction (CMAE) Program (DFHMAERR) Applid (CICSDEV);
Qname(CIA.SYSTEM.ERROR.QUEUE); CompCode = 2 Reason = 216
```

would use this error message format:

```
05  WS-WRITE-ERRORQ-ERROR.
    10  WS-WRITE-MESSAGE      PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(02)  VALUE SPACES.
    10  WS-WRITE-MQI-VERB     PIC X(06)  VALUE SPACES.
    10  FILLER                PIC X(40)  VALUE
         ' error reported in CICS SFR transaction('.
    10  WS-WRITE-TRX          PIC X(04)  VALUE SPACES.
    10  FILLER                PIC X(11)  VALUE
        ')  Program('.
    10  WS-WRITE-PROGRAM      PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(10)  VALUE
        ')  Applid('.
    10  WS-WRITE-APPLID       PIC X(08)  VALUE SPACES.
    10  FILLER                PIC X(09)  VALUE
        '); Qname('.
    10  WS-WRITE-QNAME        PIC X(48)  VALUE SPACES.
    10  FILLER                PIC X(14)  VALUE
        '): CompCode = '.
    10  WS-WRITE-COMPCODE     PIC -9(9)  VALUE ZEROES.
    10  FILLER                PIC X(11)  VALUE
        '  Reason = '.
    10  WS-WRITE-REASON       PIC -9(9)  VALUE ZEROES.
```

See the *WebSphere MQ Application Programming Reference* for an explanation of completion and reason codes.

Glossary

This glossary contains terms pertaining to both CICS Service Flow Runtime and Service Flow Modeler.

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

activity

In BTS, one part of the process managed by CICS business transaction services. Typically, an activity is part of a business transaction.

Activities can be hierarchically organized, in a tree structure. An activity that starts another activity is known as a parent activity. An activity that is started by another is known as a child activity.

A program that implements an activity differs from a traditional CICS application program only in its being designed to respond to BTS events.

adapter service

An adapter service is the generated output of the Service Flow Modeler. An adapter service is a reusable composed business function that exposes a programmatic interface to a service requestor in an Enterprise Information System. The adapter service is deployed to, and works with, Service Flow Modeler-supported runtime environments, one of which is the CICS Service Flow Runtime V3.1. The adapter service can have characteristics that can support both metadata and generated code deployments. The adapter service has port bindings specific to each potential *deployment environment*. WSDL binding information can be extended to save any of the additional metadata required to support application code generation or other deployment and runtime behavior. The adapter service contains business services composed from a set of supported connector flow services. The adapter service is composed from runtime patterns with varying degrees of complexity and persistence. The level of abstraction needed by a given customer for the business operations will determine if connector flows are sufficient or if connector flows all driving the same EIS, need to be combined into adapter services.

Depending on how one models the adapter service that will be deployed to the CICS Service Flow Runtime V3.1 environment, it can contain a wide variety of functionality, such as control flow, data flow, sequential navigation, conditional branching including decision and iteration, data typing, storing data context, transformation of data elements, logical operations and custom code.

See the Service Flow Modeler help, available in the WebSphere Developer for System z infocenter, for information on how to model, generate and deploy an Adapter service.

adapter service request

The means by which the service requestor invokes the CICS Service Flow Runtime. An Adapter service request is sent in the form of a request message.

adapter service request processing

© Copyright IBM Corp. 2001, 2010

The programmatic functions (modeled at build time using the Service Flow Modeler tool) that an adapter service performs in order to manage and fulfill a business transaction on the server run time. To handle the work required by adapter request processing, CICS Service Flow Runtime can invoke one or multiple server adapters without requiring action by the service requestor. Each adapter service request results in a different instance of the Navigation Manager, Navigators (sometimes but not always) and only those server adapters that are needed to support that adapter service request.

adapter service response message
Sometimes referred to a reply message, this is a message sent out of the CICS Service Flow Runtime in response to an adapter service request that is sent from the service requestor. An adapter service response message contains the result of processing the business transaction that was defined in the request message. Not every adapter request message merits a reply. At build time the request message is formatted to indicate whether or not a reply is required.

The adapter service response message is an application-level reply. It is different from a response that is required by a communications protocol. For example, EXCI requires that all requests be responded to at the protocol level. Therefore, if the service requestor used EXCI for the adapter request message and if no adapter reply message was required, a protocol-level response would still be sent. However, this protocol-level response would not be performed by the CICS Service Flow Runtime. This protocol-level response would not have to be addressed during build time, diagnostics and tracing.

asynchronous
An event that occurs at a time that is unrelated to the time at which another event occurs. The two events are mutually asynchronous. The relationship between the times at which they occur is unpredictable.

asynchronous mode
A type of CICS Service Flow Runtime processing in which the BTS process implements an instance of the CICS Service Flow Runtime is run asynchronously from the initiating unit-of-work. All BTS activities within that BTS process will be run asynchronously from their parent activities. This has the effect of running the BTS process and all activities as separate units-of-work each with a distinct commit scope.

You would typically want to process a request in asynchronous mode if as a result of the processing, data will be updated.

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.

audit trail utility
A CICS-supplied utility program, DFHATUP, that enables you to print selected BTS audit records from a specified logstream.

authentication
In computer security, verification of the identity of a user or the user's eligibility to access an object. In CICS Service Flow Runtime, the authentication process is established within the WebSphere MQ-CICS bridge via an AUTH parameter passed to the bridge monitor at startup.

B

basic mapping support (BMS)
Basic Mapping Support (BMS) is a design component of a CICS application that handles the presentation logic of the CICS transaction and relieves the application developer from having to encode and decode 3270 terminal data streams. BMS is an application programming interface between CICS programs and terminal devices. A BMS map set is made up of maps that specify how field data is to be formatted. At build time, an Application developer can use the BMS importer function in Service Flow Modeler to import a BMS map set from an application on the EIS into the workbench studio. An application developer can also assign the actual BMS field names to fields on 3270 application screens that he or she has imported. Using the 3270 screen importer you can add BMS to non-BMS CICS target application screens. At run time, CICS Service Flow Runtime V3.1 uses the compiled BMS source code in the form of a load module to determine field information such as type and length as well as extended attributes. Application screens that are real BMS would already have this load module. This feature allows developers to create the BMS-equivalent for non-BMS CICS screens.

build time
The time period when business transaction processing is defined, modeled or modified electronically. At build time, a programmer that is familiar with the enterprises business processes uses the Service Flow Modeler tool to:

- capture existing EIS interfaces
- model a newly composed business service using these interfaces
- generating a new adapter
- deploy the adapter as a service to the supported runtime environment

build time environment
A modeling environment. The Adapter service modeling environment for the CICS Service Flow Runtime is provided by the Service Flow Modeler, which is an eclipse-based application integration plug-in to the WebSphere Developer for System z product.

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. An account transfer for example. Traditionally, in CICS a business transaction might be implemented as multiple user transactions. Using BTS, a business transaction might be implemented as multiple activities. In CICS Service Flow Runtime, the adapter service that was modeled, generated and deployed to the runtime environment enables the processing that will manage and complete the business transaction.

C

CICS (Customer Information Control System) is an online transaction processing (OLTP) program from IBM that, together with the COBOL programming language, has formed over the past several decades the most common set of tools for building customer transaction applications in the world of large enterprise mainframe computing. A great number of the legacy applications still in use are COBOL/CICS applications. Using the application programming interface (API) provided by CICS, a programmer can write programs that communicate with online users and read from or write to customer and other records (orders, inventory figures, customer data, and so forth) in a database (usually referred to as "data sets") using CICS facilities rather than IBM's access methods directly. Like other transaction managers, CICS can ensure that transactions are completed and, if not, undo partly completed transactions so that the integrity of data records is maintained.

message header
The required portion of the adapter request message that provides the meta information used by the CICS Service Flow Runtime for the processing of a message in CICS.

CICS transaction
A unit of application data processing (consisting of one or more application programs) initiated by a single request, often from a terminal. A transaction may require the initiation of one or more tasks for its execution.

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 communicates with a given terminal. The area can also be used to pass data between programs within a task. At run time, the DPL Stub program (DFHMADPL or DFHMADPP) requires that information from the service requestor be passed in the form of a communication area.

compensation
The act of modifying the effects of a child activity. Typically, compensation undoes the actions taken by an activity. For example, compensation for an order activity might be to cancel the order. Compensation is taken under consideration at build time. If a business transaction is to include compensation, the adapter service model needs to reflect it. It can modify the effects of many activities within a given process.

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.

Examples of a connector include the following:

- Host On Demand (HOD) - 3270 or 5250 connector style
- CICS Transaction Gateway (CTG) – data structure / transactional style
- Java Messaging Service WebSphere MQ – asynchronous Messaging style

- JCA connectors for these interfaces, IMSConnect interfaces and JCA connectors, runtime modules produced with CICS Service Flow Runtime, and CICS's FEPI, DPL, Link3270, and Web Bridge
- SOAP listener.

custom program
A program that augments adapter service request processing. A custom program can contain complex rules such as logic and complex I/O that could not be modeled using the Service Flow Modeler. To invoke the custom program is exactly the same as the mechanism to invoke sequence flow processing with CICS transactions, (that is, DPL.)

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 as long as they all have different names. A data-container can be read by all the activities that comprise a process.

deployment
The word "deploy" means to place files or install software into an operational environment. In J2EE, this involves creating a *deployment descriptor* suitable to the type of application that is being deployed. In the case of Service Flow Modeler, the deployment descriptor would define the components and operating system parameters of the Adapter service. Service Flow Modeler will be one of multiple tools the WebSphere Developer for zSeries product that allows for multiple *deployment environments* that are required to create the programmatic interfaces to Enterprise Information Systems. In Service Flow Modeler, a deployment for existing EIS may require both native, non-Java and standard Java deployments (such as HATS) to maintain Quality of Service (QoS), skills and existing IT investment in target EIS environments Deployment properties for the Adapter service are set at generation time.

deployment descriptor
Deployment descriptors are standard text files, formatted using XML notation and you package them in the application that you are deploying. Deployment descriptors define components and operating parameters for the application that you are deploying. In Service Flow Modeler, the deployment descriptor describes the Adapter service implementation.

deployment pattern
Sometimes referred to as an *access pattern*, a deployment pattern refers to way in which an Adapter service (composed using the Service Flow Modeler) complies with a set of well defined usage patterns for execution in a supported runtime environment. As part of the sequence of tasks associated with modeling an Adapter service, the developer sets properties that specify the Adapter service deployment pattern. The deployment pattern defines a combination of elements or characteristics that are passed to the CICS Service Flow Runtime when the Adapter service is generated and deployed. Deployment patterns describe how the Adapter service behaves on the supported runtime environments.

Deployment patterns vary depending on the characteristics of the Adapter service and the nature of the request. For example, an Adapter service of a complex business function may require access to multiple target

applications and may result in data being updated. While an Adapter service of a simple business function may require access to only a single target application and may result in no data being updated (a simple inquiry, for example). The differences between simple and complex deployment patterns manifest at run time, as the CICS Service Flow Runtime performs the necessary program processing to handle the specific deployment patterns.

Distributed Program Link (DPL)
A function of CICS intersystem communication that enables CICS to ship LINK requests between CICS regions. CICS Service Flow Runtime can initiate programs, including custom programs using one or a sequence of DPLs, via CICS LINK.

E

EIS interface
An EIS interface will vary from organization to organization and its nature will be determined by the systems and application architecture. From the perspective of the Service Flow Modeler user, the EIS interface represents the data source on which the user will focus his or her development efforts. Many enterprise information systems have interfaces that do not lend themselves to participation in SOA. Developers are able to use Service Flow Modeler to model and compose the EIS interface (5250 and 3270 screens, COBOL record descriptions, transactions) to 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 SOA.

For information on how developers work with and manipulate the artifacts that comprise an existing EIS interface to create, generate and deploy a service of an existing EIS application, see the Service Flow Modeler help in the WebSphere Developer for z/Series information center.

Enterprise Information System (EIS)
The applications that comprise an enterprise's existing mission critical system(s) for handling company-wide information. These applications provide an information infrastructure for an enterprise. An enterprise information system offers a well defined set of services to its clients. These services are exposed to clients as local and/or remote interfaces. Examples of enterprise information systems include:

- enterprise resource planning systems
- mainframe transaction processing systems
- legacy database systems

Many organizations are now tasked with evolving existing application infrastructure to a modern service oriented architecture (SOA).

error logging
the process of writing error information to a file.

F

FEPI Front End Programming Interface. A terminal emulator that permits CICS programs to interact with other 3270-based applications through virtual terminal sessions. In CICS Service Flow Runtime, a server adapter can interface with IBM's FEPI as part of processing a business transaction. The server adapter interaction with FEPI must be modeled and defined at build time. Using IBM's FEPI product, the server adapter can send requests to

and receive replies from any CICS and IMS application whose 3270 datastream is intended for a SLU2 3278 Model 2 terminal (24 rows by 80 columns), that is, the returned buffer in the a single send and receive is not greater than 3600 bytes.

flow Sometimes referred to as a sequence flow, a flow is a graphical representation of the sequence of activities performed in accordance with the business processes of an enterprise. Flows consist of a graph of nodes, with defined entry and exit points. Each node represents invoking a service operation, controlling the flow of the sequence, or performing reusable business logic. Flows are exposed as a service themselves, to be driven externally.

At build time, developers can use the flow editor component of the Service Flow Modeler to compose a visual model of an Adapter service. The flow illustrates the sequence of screens and the flow of information of an existing EIS application, for which a service is being created.

H

HATS (Host Access Transformation Services) IBM software set of tools that provides Web-based access to legacy data sources and host-based applications. Using Service Flow Modeler, an enterprise developer can deploy an Adapter service to multiple supported runtime environments, including Host Access Transformation Services I/O on WebSphere Application Server.

host In CICS Service Flow Runtime, the host is the computer system (such as a mainframe transaction processing system) on which EIS applications may reside. Developers can use components of the Service Flow Modeler to access applications that run on a host and import resources (such as screens and BMS maps) into the build time environment. Developers can then manipulate the imported resources to create, generate and deploy service-like interface of the host application.

host application
An application residing on the host computer system.

host connection properties file
The resource in a Service Flow Modeler project that contains the information necessary for connecting to the host system during build time. The host connection information that is stored on the host connection properties file enables you to access the host system for the purpose of capturing screens and recording dialogs. The host connection properties file also stores configuration properties that define the connection to the Enterprise Information System (EIS) at run time.

I

importer
Importers are a set of components in the Service Flow Modeler that are used to populate parts of the information model from existing resources. These resources may 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. For format definitions and screen captures, when the resource is imported, a schema is created. When navigation information is imported, flow info, WSDL and, if possible, schema are created. Data format importers are specific to the language the

resources are written in and screen format importers are specific to the application environment the formats are for.

inline DPL
Inline DPL refers to the use of a DPL command within a FEPI sequence flow modelled using the Service Flow Modeler. 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 and/or receives. It does this 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:

- By CICS to implement user-defined resource protection (logging to the system log).
- By CICS to implement user-defined automatic journaling (to any journal, including the system log) .
- Explicitly by the JOURNAL command from any application program (user journaling to any journal, including the system log).

journaling
The recording of information onto any journal (including the system log), for possible subsequent processing by the user. The primary purpose of journaling is to enable forward recovery of data sets. In CICS Service Flow Runtime, journaling refers to the collecting and maintaining information about the state of the runtime environment and application data to enable the compensation and recovery of the processing of an adapter request message.

The runtime journaling facility uses CICS/BTS container services to support compensation. Journal information is maintained only during the processing of each adapter request message, except in the case of failure. In the case of failure, CICS Service Flow Runtime retains state information and application data for subsequent use in a compensation flow.

The Navigation Manager, Navigators and server adapters participate in capturing two types of data that are used for compensation:

- State information is stored in the process and status data-containers as part of the BTS process.
- Journal data is stored in the journal data-container as part of the BTS process.

M

message
A set of data that is passed from one application to another. In Service Flow

Modeler, a message can be modeled by a *message definition file* that describes the structure and content of the message. Messages must have a structure and format which is agreed by the sending and receiving applications.

N

Navigator

CICS Service Flow Runtime programs that perform adapter request processing, manage states during the microflow processing and invoke server adapters. The Navigator and the server adapters are generated as the result of modeling via the Service Flow Modeler.

Not every Adapter service created with Service Flow Modeler requires a Navigator. In Adapter services of the single connector type, there is only the Navigation Manager product program, DFHMAMGR, and a single generated and deployed server adapter program. There is no generated and deployed adapter Navigator program since there is no need to control and manage the flow as is the case when multiple server adapter programs (aggregate connector type).

Navigation Manager

CICS Service Flow Runtime program that invokes the Navigator programs. Runs as DFHROOT in BTS process.

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 may restrict the actual parameters that are possible. EIS operations are generally not independent of each other. Only certain sequences are possible and the business client end of the conversation must be kept active for the duration of the conversation. This implies that the developers view of stateless connections and tasteful ones should not be the same. Operations on a stateless connection are complete operations like those implemented by an EIS service.

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. In Service Flow Modeler, certain runtime patterns will allow and adapter service to be can categorized persistent or nonpersistent. In Service Flow Modeler, the term persistent characterizes whether or not the transactional instance data will be persist or remain in a BTS repository data set for the named process under which the Adapter service is executing. A developer would want to make an adapter nonpersistent if he or she were not concerned with maintaining data should the process under which the adapter is running fails. Session state information is a good example of persisted state data.

CICS Service Flow Runtime supports persistence in the processing of adapter services of both the single connector and the aggregate connector type.

process
In BTS, a collection of one or more activities. A process is the largest unit that CICS BTS can work with and has a unique name by which it can be referenced and invoked. In CICS Service Flow Runtime, the process is uniquely identified by the 36 byte process name value in the message adapter message header (DFHMAH).

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 requestor to the CICS Service Flow Runtime to invoke an Adapter service to process a business transaction. If the service requestor is WebSphere MQ-enabled, the adapter request message is of the form of an WebSphere MQ message. If the service requestor is using a CICS-supplied interface the request message is of the form of a communication area (COMMAREA).

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.

runtime environment
The supported environments to which the Adapter services, modeled and generated using the Service Flow Modeler are deployed.

run time
The time period during which the Adapter service instance is operational, with business transactions being managed and completed in the application server.

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 within 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. In Service Flow Modeler a screen represents a schema or message that corresponds to a known terminal screen structure. Keep in mind that a single screen can have more than one state. Using Service Flow Modeler, you can address multiple states of a single screen by assigning a description that corresponds to each screen state. Therefore, a single screen may have multiple screen descriptions.

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 would appear at known screen locations if the second program was being accessed by a human operator. The FEPI server adapter performs screen navigation to capture 3270 screen images from CICS and IMS applications.

sequence flows
In the Service Flow editor, a graphical representation of a composed service. A sequence flow is composed of a sequence of operations, assignments, and conditionals that are wired together into finite paths such that a request message is processed resulting in a response message. A sequence flow has a WSDL operation interface. A sequence flow directs actions from a beginning point (Receive node) through a number of operations to an end point (Reply node). A sequence flow can be representative of and existing application running on an enterprise information system (EIS). In Service Flow Modeler, the sequence flow represents the re-purposing of the EIS application as a service. See *flow*.

server adapters
Any one of three types of programs in the server run time that are invoked by the Navigator program to perform the business transaction activity defined within a sequence flow at build time.

Server adapters include the following:

- The FEPI server adapter that interfaces to the CICS and IMS applications. It performs screen navigation. It also supports inline DPL commands.
- The CICS server adapter that interfaces to the existing CICS transactions, including custom programs that can be developed to augment the Message Adapter, via DPL.
- The WebSphere MQ server adapter that interfaces to the WebSphere MQ-enabled applications.
- The Link3270 server adapter that interfaces to CICS applications. Typical 3270 application programs use CICS Basic Mapping Support (BMS) SEND MAP and RECEIVE MAP commands to communicate with the terminal user. The BMS commands reference the BMS symbolic map, which contains field data. These programs can be accessed using a generated Link3270 server adapter, with the CICS Link3270 bridge mechanism as modelled and deployed via the Service Flow Modeler at build time.

service
An external entity outside the organization that affects the organization's processes.

Service Flow Modeler
An eclipse-based application integration toolset that is targeted to the Application Transformation space. Service Flow Modeler is delivered as a developer plug-in to the WebSphere Developer for zSeries product. The Service Flow Modeler represents a single set of core components that enable the developer to:

- capture existing EIS interfaces
- model a newly composed business service using these interfaces
- generate a new adapter
- expose the new adapter as a service.

The end result enables the organization to expose their existing applications as a service-like interface, facilitating the move to service oriented architecture (SOA).

service provider
The business owner of the service. Technically, the service provider is the application that hosts access to the service. A service provider describes its service using WSDL. This definition is published to a directory of services. The directory could use Universal Description, Discovery, and Integration (UDDI). Other forms of directories can also be used.

service requestor
The business that requires certain function to be satisfied. Technically, a service requestor is the application that is looking for and invoking or initiating an interaction with a service). The requestor role can be played by a browser driven by a person or a program without a user interface, e.g. another Web service. A service requestor issues one or more queries to locate a service and determine how to communicate with that service.

T

target application
The target application is the CICS or IMS application that contains the data or information required to fulfill the business transaction. Using the Service Flow Modeler, a developer can model an adapter that uses various interfaces / access patterns to CICS/IMS target applications. The runtime also supports passthrough processing as a way to access CICS target applications. Passthrough processing does not require developers to use the Service Flow Modeler. Regardless of how developers choose to access the target application, the data that is retrieved from the target application is sent back in a response message to the service requestor that made the request.

transaction
A transaction is the controlled interaction between two entities, usually involving the passing of information. Transactions enforce ACID properties (atomicity, consistency, isolation, and durability), and in certain cases transactions can be rolled back, or reversed to a certain point. A unit of processing consisting of one or more application programs initiated by a single request. A transaction can require the initiation of one or more tasks for its execution.

transform
The process of changing the structure and values of data from one form to another. Service Flow Modeler transforms or “adapts” existing interfaces on an EIS in order to facilitate participation of EIS applications in a service in an SOA.

transform connectivity
An application transformation style that enables customers to change the way in which enterprise applications are accessed and used. Using Service Flow Modeler, one can transform connectivity by exposing existing applications as a service-like interface, facilitating the move to service oriented architecture (SOA).

W

Web Services Definition Language (WSDL)
The standard format for describing a web service. An XML format for

describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. A WSDL definition describes how to access a web service and what operations it will perform.

#

3270 The 3270 Information Display System, a product from IBM, was, prior to the arrival of the PC, the way that almost the entire corporate world interfaced with a computer. Many thousands of corporate legacy application programs written to interact with users at 3270 terminals are being used from PCs equipped with software known generally as 3270 emulators. TN3270 is a program that provides PC users remote (Telnet) connection to an IBM computer that is running 3270 applications. At build time, a developer can use Service Flow Modeler to connect to legacy application programs and to import resources (screens, COBOL record descriptions) into a project created in the WebSphere Developer for zSeries workbench.

Index

A

Abend errors
  processing 240
adapter
  examples 5
adapter service
  deploying 63
  updating 65
Aggregate connector
  nonpersistent 54
  persistent 54

B

Business Transaction Services
  data-containers 114

C

checklist for troubleshooting 202
Code page
  conversion
    support statement 76
Compensation
  about 194
  data 195
Complex deployment pattern 54
container
  DFHMAC-ALLPARMS 76
  DFHMAC-ERROR 77
  DFHMAC-LNK3270V1 78
  DFHMAC-PASSTHRU 78
  DFHMAC-REQUESTV1 78
  DFHMAC-SYSPARMV1 79
  DFHMAC-USERDATA 79
  DFHWS-DATA 79
containers
  using 71
Conversion
  code page 76

D

Data conversion
  description 74
  DFHCNV
    customization 75
  templates 75
    sample 316
deployment patterns
  complex 54
  passthrough 54
  simple 54
DFHCNV
  customization
    description 75
DFHMAC-ALLPARMS container 76
DFHMAC-ERROR container 77
DFHMAC-LNK3270V1 container 78
DFHMAC-PASSTHRU container 78
DFHMAC-REQUESTV1 container 78
DFHMAC-SYSPARMV1 container 79
DFHMAC-USERDATA container 79
DFHMAH
  structure 80
DFHMAH2
  structure 87
DFHWS-DATA container 79
diagnosing problems 199
Distributed Programming Link (DPL) 8
dumping
  properties file 99

E

Error
  messages
    0100s 220
    0133s 224
    0200s 226
    0300s 227
    0400s 228
    0500s 231
    0600s 233
    0700s 237
    0800s 240
    0810s 244
    0820s 248
    0830s 250
    abend 237, 240
    ACQUIRE-PROCESS-ERRMSG 235
    ALLOCATE-ERRMSG 228
    BTS 233
    CANCEL-PROCESS-ERRMSG 235
    CHECK-ACTIVITY-ERRMSG 235
    CHECK-PROCESS-ERRMSG 234
    CIAIxxxx 255
    CICS API 244
    CICS XML parse 250
    CICS-ASSIGN-ERRMSG 246
    CICS-DEQUEUE-ERRMSG 247
    CICS-ENQUEUE-ERRMSG 246
    CICS-INQUIRE-ERRMSG 247
    CICS-LEVEL-ERRMSG 242
    CICS-RECEIVE-ERRMSG 244
    CICS-RETRIEVE-ERRMSG 245
    CICS-START-ERRMSG 245
    COMMAREA-ERRMSG 241
    data-container 226
    DEFINE-ACTIVITY-ERRMSG 233
    DEFINE-EVENT-ERRMSG 236
    DEFINE-PROCESS-ERRMSG 233
    DELETE-CONTAINER-ERRMSG 227
    DFHL3270-BRIH-ERRMSG 237
    DFHL3270-ERRMSG 237

© Copyright IBM Corp. 2001, 2010

Error *(continued)*
messages *(continued)*
DPL server adapter 227
DPL-ERRMSG 227
EIBCALEN-ERRMSG 241
EXTRACT-ERRMSG 229
FEJBDTRN/E-ERRMSG 243
FEPI server adapter 228
FILE-DELETE-ERRMSG 224
FILE-ENDBR-ERRMSG 223
FILE-READ-ERRMSG 220
FILE-READNXT-ERRMSG 222
FILE-RECTYPE-ERRMSG 221
FILE-REWRITE-ERRMSG 222
FILE-STARTBR-ERRMSG 221
FILE-WRITE-ERRMSG 223
FREE-ERRMSG 229
GET-CONTAINER-ERRMSG 226
INQUIRE-ERRMSG 230
INQUIRE-TRANSID-ERRMSG 239
INVALID-ATTRIBUTE-WARNING 239
INVALID-FORMAT-ERRMSG 242
ISSUE-ERRMSG 230
Link3270 237
MAP-NOT-FOUND-ERRMSG 239
MAP3270-ERRMSG 230
MAPSET-LOAD-ERRMSG 239
miscellaneous 240
MQ call error 231
MQCLOSE-ERRMSG 232
MQGET-ERRMSG 232
MQOPEN-ERRMSG 231
MQPUT1-ERRMSG 231
NO-MAPNAME-ERRMSG 237
NO-VECTOR-ERRMSG 238
PASSTICKET-ERRMSG 228
POST-ERR-ERRMSG 244
PROPERTY-ERRMSG 230
PROTECTED-UPDATE-WARNING 238
PUT-CONTAINER-ERRMSG 227
RECEIVE-ERRMSG 229
RETRIEVE-EVENT-ERRMSG 236
RUN-ACTIVITY-ERRMSG 234
RUN-PROCESS-ERRMSG 234
SEND-ERRMSG 228
SET-CONTAINER-ERRMSG 226
SET-ERRMSG 229
TDQ-READ-ERRMSG 242
TDQ-WRITE-ERRMSG 241
temporary storage queue 224
TS-DELETE-ERRMSG 225
TS-INQEND-ERRMSG 226
TS-INQNEXT-ERRMSG 226
TS-INQSTART-ERRMSG 226
TS-READ-ERRMSG 225
TS-REWRITE-ERRMSG 225
TS-WRITE-ERRMSG 225
UNEXPECTED-VECTOR-ERRMSG 238
UNSUPPORTED-CICS-ERRMSG 240
UNSUPPORTED-VECTOR-ERRMSG 240
update properties file 248
Error *(continued)*
messages *(continued)*
VALIDATION-ERRMSG 242
VSAM file 220
XML-CONVERT-ERRMSG 250
XML-FIELD-LEN-ERRMSG 252
XML-INVALID-TAG-ERRMSG 252
XML-NO-FIELD-ERRMSG 252
XML-NO-MAP-ERRMSG 251
XML-NO-SCREEN-ERRMSG 251

F

Facility assignment processing
non-unique UserIDs
Link3270 bridge server adapter 366
Facility state cleanup processing
description 168
TSQ
description 169
VSAM
description 169
FEPI
inline DPL 155
LU assignment processing 158
unique IDs synchronous mode 160
server adapter processing
LU assignment processing 158
File dumps
Error file 283
Properties file 272
Files
DFHMAC1F
description 111
DFHMACOF
description 111
DFHMAERF
description 112
DFHMAL2F
description 112
DFHMALRF
description 114
DFHMALSF
description 113
DFHMALVF
description 114
DFHMAMPF
description 113
DFHMATIF
description 111
error 372
run time
descriptions of 110
Error file 110
FEPI (SLU) Connection alternate file 110
FEPI (SLU) Connection file 110
FEPI Target Interaction file 110
Link3270 repository 110
Link3270 State 110
Properties file 110

I

inline DPL 155
Installation
  customization
    Link3270 facility state cleanup 30
    PLT processing 30
installation verification procedure
  messages 253
Installation Verification Procedure
  adapter programs 319
  back-end transactions 318
  copybooks 319
  mapset 319
  resource definition JCL 320
  sample listing
    back-end transactions 318
invoking adapter services 67

J

JCL
  IVP back-end transactions 318, 319
Jobs
  customizing the build time templates 28
Journaling
  about 194

L

Link3270 bridge
  Facility assignment processing 366
LU assignment processing
  non-unique UserIDs
    FEPI adapter 158
    synchronous mode 160

M

Map Header (CIA-MAP-HEADER)
  field definitions
    CIAMH-CURSPOS-FIELDNAME 96
    CIAMH-DATA-FORMAT-INDICATOR 95
    CIAMH-EXT-ATTRIBUTE-COUNT 96
    CIAMH-EXT-ATTRIBUTES 96
    CIAMH-FIELD-COUNT 96
    CIAMH-MAP-COLUMNS 96
    CIAMH-MAP-DATA-LENGTH 95
    CIAMH-MAP-NAME 96
    CIAMH-MAP-ROWS 96
    CIAMH-MAPSET-NAME 96
    CIAMH-STRUCID 95
message
  request
    non-passthrough header 80
    passthrough header 87
    passthrough map header 94
    passthrough screen header 93
Message Header (DFHMAH)
  field definitions 81
    DFHMAH-ABENDCODE 82
Message Header (DFHMAH) *(continued)*
  field definitions *(continued)*
    DFHMAH-COMPCODE 82
    DFHMAH-CORRELID 86
    DFHMAH-DATALENGTH 84
    DFHMAH-FAILED-NODE 86
    DFHMAH-FAILED-PROCNAME 85
    DFHMAH-FAILED-PROCTYPE 85
    DFHMAH-FAILED-PROGRAM 86
    DFHMAH-FAILED-TRANID 85
    DFHMAH-FORMAT 81
    DFHMAH-LINKTYPE 86
    DFHMAH-MESSAGE 82
    DFHMAH-MODE 82
    DFHMAH-MSGID 86
    DFHMAH-PROCESSNAME 83
    DFHMAH-PROCESSTYPE 83
    DFHMAH-REPLYTOQ 85
    DFHMAH-REPLYTOQMGR 85
    DFHMAH-REQUESTNAME 84
    DFHMAH-RETURNCODE 82
    DFHMAH-STRUCID 81
    DFHMAH-STRUCLENGTH 81
    DFHMAH-SUSPSTATUS 82
    DFHMAH-UOWCONTROL 82
    DFHMAH-VERSION 81
Message Header (DFHMAH2)
  field definitions
    DFHMAH2-APPLRESPTIME 92
    DFHMAH2-CLIENTIP-ADDRESS 92
    DFHMAH2-DATALENGTH 89
    DFHMAH2-DEALLOCATE-IND 91
    DFHMAH2-FACILITYKEEPTIME 90
    DFHMAH2-FACILITYLIKE 90
    DFHMAH2-FORMAT 89
    DFHMAH2-GETWAITINTERVAL 91
    DFHMAH2-INITIAL-AID-BYTE 91
    DFHMAH2-NEXT-TRANSID 90
    DFHMAH2-RECEIVE-TYPE 90
    DFHMAH2-RESERVED2 92
    DFHMAH2-RESPTIME 92
    DFHMAH2-SEND-AID-FIRST 91
    DFHMAH2-STRUCID 88
    DFHMAH2-STRUCLENGTH 89
    DFHMAH2-TRANSID 90
    DFHMAH2-USE-FKEEPTIME-IND 90
    DFHMAH2-VECTOR-LOGGING 91
    DFHMAH2-VERSION 89
  passthrough
    field definitions 88
messages 255
MODE parameter 63, 65

N

Non-unique userID
  processing 177
    FEPI adapter 158
    Link3270 bridge server adapter 366
Non-unique UserIDs
  FEPI adapter 158

Non-unique UserIDs *(continued)*
Link3270 bridge server adapter 366

P

passthrough deployment pattern 54
PLT
configuration
Link3270 facility state cleanup 30
post-installation 255
problem determination
recommended documentation 199
Problem determination
procedures
using the Error file dump 207
using the Error file dump 207
Process containers
about 115
single connector patterns 121
processing
Web services 176
Processing modes
asynchronous 55, 127
synchronous 55, 127
processing XML 183
Programs
error listener 372
IVP back-end transactions 318, 319
properties file
dumping 99
overview 65
updating 63

R

request message
headers
CIA-MAP-HEADER 94
CIA-SCREEN-HEADER 93
DFHMAH 80
DFHMAH2 87
run time
invoking 67
Run time
invoking
using WebSphere MQ triggering 69
processing modes 127
runtime environment
overview 3

S

Screen Header (CIA-SCREEN-HEADER)
field definitions 93, 95
CIASH-ATTENTIONID 93
CIASH-MAP-COUNT 94
CIASH-STRUCID 93
Selecting
processing modes 55
server adapter
DPL 8
server adapter *(continued)*
Web service 10
Service Flow Modeler
description
editors 5
importers 5
Service requester
description 67
interfaces 67
Simple deployment pattern 54
Single connector
nonpersistent 54
persistent 54
State management
description 179

T

trace points 213
troubleshooting
checklist 202

U

using containers 71
utilities
error file dump 100
Link3270 vector log file dump 100
properties file dump 100
properties file update 100

V

validating DFHMAINJ 24

W

Web services 10
processing 176

X

XML processing 183

Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information 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 right 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 information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
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 states 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 information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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 wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

© Copyright IBM Corp. 2001, 2010

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

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 information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

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.

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

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

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  IBM United Kingdom Limited
  User Technologies Department (MP095)
  Hursley Park
  Winchester
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–816151
  - From within the U.K., use 01962–816151
- Electronically, use the appropriate network ID:
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

© Copyright IBM Corp. 2001, 2010

SC34-5899-10

Spine information:

CICS Service Flow Runtime User's Guide Version 3 Release 1