CICS Transaction Server for z/OS
Version 5 Release 6

Event Processing in CICS

Note

Before using this information and the product it supports, read the information in Product Legal Notices.

This edition applies to the IBM® CICS® Transaction Server for z/OS®, Version 5 Release 6 (product number 5655-Y305655-BTA) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this PDF

This PDF provides introductory and guidance information on how to use events to monitor the enterprise and its business, and react accordingly.

Note: CICS TS 5.4 introduces support for system rules in CICS policies, which provide equivalent or enhanced capability over system events. As a result, no further enhancements will be delivered to system events in future CICS releases, so consider migrating to CICS policies at the earliest opportunity.

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

Date of this PDF

This PDF was created on October 11th 2019.

Chapter 1. Event processing overview

An event is anything that happens that is significant to an enterprise. Event processing is the capture, enrichment, formatting and emission of events, the subsequent routing and any further processing of emitted events (sometimes in combination with other events), and the consumption of the processed events.

Events provide a way to notify users or other software products about activities that are occurring in your CICS applications and systems. If you are planning how to use events, or want more information about event processing in CICS, this section provides a summary of the technologies and concepts to help you get started.

Events can be produced throughout a business enterprise. At the edges of the enterprise, events can be detected by sensors. In the enterprise network, events can be produced when business processes start and complete or fail. The activity of the enterprise and its business can be monitored and changed as a result of events.

Event processing architecture

An event processing architecture is based on interactions between three components: an event source, an event processor, and an event consumer.

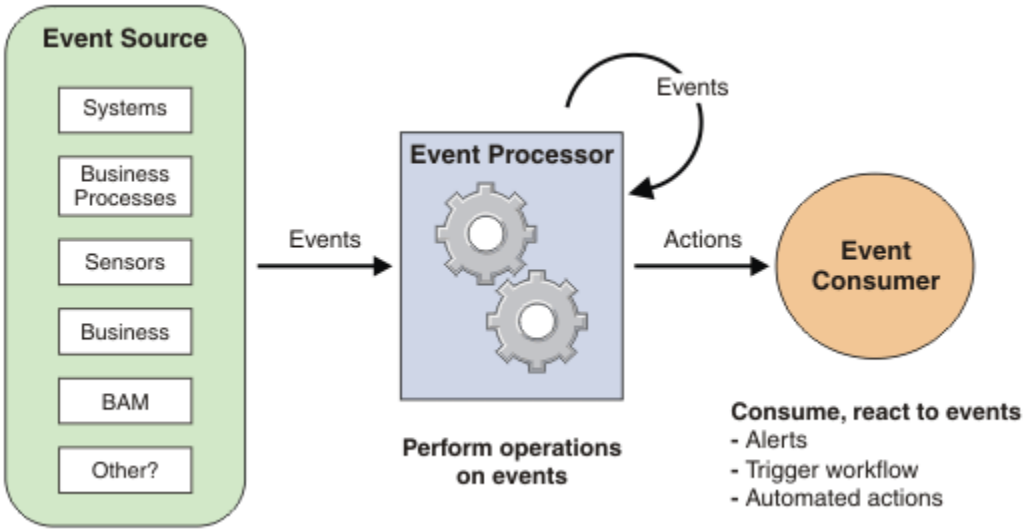


Figure 1. Event processing architecture

Event source

An event source emits events into the event processing system. Examples of event sources are simple Radio Frequency Identification (RFID) sensors and actuators, business flows, CICS applications, and CICS system components.

Event processor

The event processing system can perform various actions on events:

- Enrich a single event in a simple manner; for example, adding a timestamp to the event data. Processing of this kind is sometimes referred to as simple event processing.
- Add information about the source of the event.

- Process multiple single events, from multiple event sources against event patterns to produce a new derived or compound event. Processing of this kind is sometimes referred to as complex event processing.

The processed event is then available to an event consumer.

Event consumer

The event consumer reacts to the event. An event consumer can be as simple as updating a database or business dashboard, or as complex as required, carrying out new business processing as a result of the event.

Here are some examples of consuming an event:

- Updating a business dashboard
- Updating a database
- Interacting with Web 2.0 components to dynamically update web pages
- Sending an alert by using email or SMS based on event data
- Putting event data on a message queue
- Starting a new piece of application work

CICS event processing provides filtering, capture, enrichment, formatting, and routing of single business events, enabling CICS to act as a source of simple business events. However, these events can be consumed by a complex event processing engine in which they can be combined with events from other sources in addition to CICS.

How event processing can help your business

CICS event processing enhances business flexibility by providing a noninvasive methodology for enhancing business applications. Events are defined and controlled independently of the business logic, extending a business application without modification. You can use event processing to enhance governance for standards compliance, and to monitor business processing.

You can use event processing to detect business situations, providing early and intelligent insight to assist you in making timely and effective business decisions. Event processing can help your business in these ways:

- Getting the correct information with the correct level of detail to the relevant person at the appropriate time.
- Facilitating quick observation of exceptional business behavior and notifying the appropriate people.
- Diagnosing problems based on symptoms and resolving them.
- Providing data for dashboard display of real-time business service availability.
- Monitoring the health of a system.

CICS event processing enhances business flexibility by providing a noninvasive methodology for enhancing business applications. Events are defined and controlled independently of the business logic, extending a business application without modification.

- CICS provides assured event emission for your business-critical, event-based solutions. Events can be used to extend applications as reliably as if you had, for example, added an MQPUT statement to an application to write an event, without actually changing the application.
- CICS event processing can emit events at a number of clearly defined points, known as capture points. A capture point is an opportunity for an event to be emitted. Capture points are provided before and after selected EXEC CICS API calls and at program start. All significant points in a CICS application can emit an event.

CICS and event processing

You can specify, capture, and emit business events from CICS. These business events can be consumed by another CICS application, or placed on an IBM MQ queue for consumption in a variety of ways, including for example, consumption by a complex event processing engine such as IBM Operational Decision Manager.

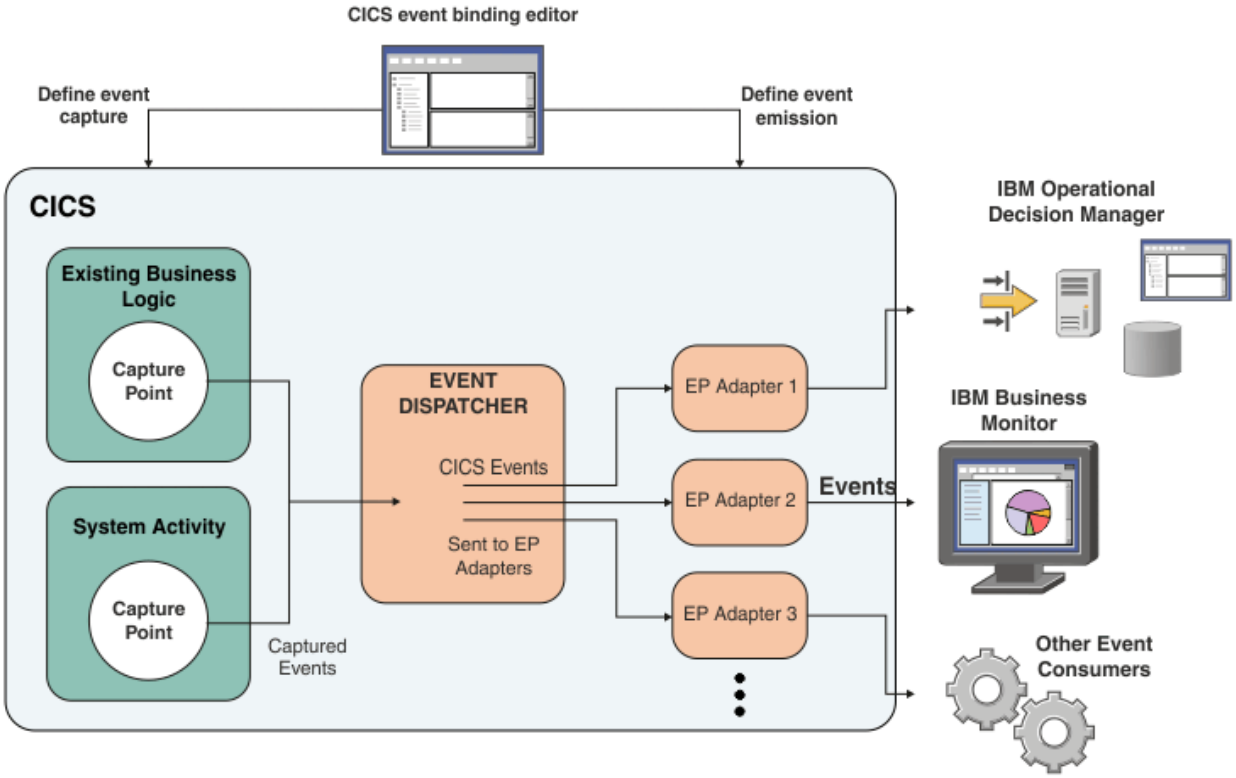


Figure 2. Event processing high level architecture

CICS Transaction Server for z/OS provides the following comprehensive support for simple business events:

- A CICS application can capture and emit events with no change to the application itself, using noninvasive capture points, before and after selected EXEC CICS API calls, and at program start.
- A CICS system can capture and emit events relating to system activity. For example, resource status changes, or crossing task thresholds.

  Note: System events support is deprecated. For equivalent function, use policy system rules instead. For more information, see Policy system rules.
- For situations in which the noninvasive capture points are not sufficient to capture a specific business event, a new EXEC CICS API call, SIGNAL EVENT, allows events to be captured anywhere in a CICS application.
- After CICS has captured an event, CICS provides a variety of EP adapters for formatting and routing events, or you can also write your own custom EP adapter.
- CICS event processing supports interoperability standards with business event consumers such as IBM Business Monitor by emitting events in the Common Base Event specification V1.01 format. The Common Base Event format is an event format used across many products.

  CICS provides support for the IBM Operational Decision Manager XML formats for events, for interoperability with components of IBM Operational Decision Manager. For details, see Developing event projects.

CICS Explorer® includes the Event binding editor, a tool that helps you to define your business events.

CICS support for event processing

CICS supports the production of events from applications when certain **EXEC CICS** commands are run, or when certain system conditions occur. For example, a file status changes or an unhandled transaction abend occurs. These events can be consumed by a variety of event consumers.

Event processing workflow

Event processing supports a flow of work between four classes of user.

The business management user
Understands the needs of the business, but does not necessarily know about the programs and computer systems that support that business. This user uses a new CICS tool to express business needs in high-level terms. For example, an online camera retail business need might be: "I need marketing to be notified of every order for a telescope that exceeds £2000." This need might exist so that the customer can be targeted with offers for high-specification digital cameras.

The application analyst or programmer
Understands, or can investigate, how an existing application was designed. The application analyst or programmer can identify which part of an application must be extended to meet the needs of the business manager. In the previous example, the application analyst or programmer must know which part of a program detects orders for telescopes, where to find the value of any such order, and where to find any other data that might be useful in satisfying the business need.

The CICS system programmer
Understands how programs interact with CICS, and can help the application analyst or programmer to deploy the new event processing resources. The system programmer can define business events to monitor the system and its applications. The system programmer can also diagnose and debug problems related to the application and its events.

The IT architect
Understands event processing in a CICS system and the benefits, requirements, and costs of using assured event emission. The IT architect is aware of the need for application users to have authority to write to the event emission recoverable transport or resource. The IT architect also understands the different combinations of emission mode (synchronous), transaction mode, and recoverable transport with regards to the EP adapter.

In your organization, some or all of these classes of user might be combined in one person. The tool used by all the users is the Event binding editor, which is part of CICS Explorer. It is designed so that the different classes of user can work as a team to implement event processing.

By using the event binding editor in CICS Explorer, you specify what events you want CICS to emit. The event binding contains:

- event specifications, which describe the data to be included in an event
- capture specifications, which define the conditions under which an event is to be captured and map the captured data to the event specification
- adapter information, which defines how the event is to be formatted and emitted

Event binding

An event binding is an XML definition that defines one or more business events to CICS. An event binding consists of event specifications, capture specifications, and adapter information.

You create an event binding in a CICS bundle project using the Event binding editor. Event bindings are included in bundles: bundles are the unit for deploying, enabling, and disabling CICS events. The event binding groups sets of event specifications that are to be handled using the same EP adapter or EP adapter set. For more information about bundles, see BUNDLE resources.

The business analyst creates an event specification. The application analyst then completes one or more corresponding capture specifications. The system programmer either completes the adapter specification embedded within the event binding or names an existing EP adapter or EP adapter set defined in a separate XML definition. For more information, see "Event processing (EP) adapter specification" on page 14. EP adapter set specification is an external EPADAPTERSET resource. For more information, see "Event processing (EP) adapter set specification" on page 14.

An event binding is saved by the Event binding editor as an XML event binding file with the suffix `.evbind` in a CICS bundle project. You can deploy the bundle to a CICS system. You define, install, enable, disable, and uninstall a deployed CICS bundle in a CICS system. For more information about deploying an event binding, see Deploying a CICS bundle in the CICS Explorer product documentation.

The Event binding editor creates an XML file that conforms to the event binding schemas. To deploy to a CICS system, an event binding must be included in a CICS bundle and must conform to the event binding schemas. The following event binding schema files are installed with CICS in `/usr/lpp/cicsts/cicsts56/schemas/eventprocessing/eventbinding`:

- `CicsEPAdapter.xsd`
- `CicsEPAdapterSet.xsd`
- `CicsEventBinding.xsd`
- `CicsEventBaseTypes.xsd`
- `CicsEventCapture.xsd`
- `CicsEventDispatching.xsd`
- `CicsEventExecCapturePoints.xsd`
- `CicsEventSpecification.xsd`
- `CicsEventSystemCapturePoints.xsd`

For more information about schemas, see Event processing schema versions in the CICS Explorer product documentation.

For more information about the Event binding editor, see Event binding editor in the CICS Explorer product documentation.

Event specification

An event specification describes an event and its emitted business information. An event specification is created in an event binding by using the Event binding editor.

The **emitted business information** defines the business data or information to be contained in the event.

Event specifications can also contain an event description. The event description can be used to provide more information about the event. For example, it might be used to indicate the intended use for this event, such as sending the data to the business orders dashboard.

An event specification also defines the business data or information to be contained in the event. The following example shows the information an event specification might contain.

Event Name: `Order_over_10000`

Event Description: `When an order in excess of $10,000 is confirmed, emit an event to a business dashboard with information about the customer that place the order.`

Emitted Business Information:

- `Order_Number (`*`numeric`*`)`
- `Order_Date (`*`numeric`*`)`
- `Customer_Reference (`*`numeric`*`)`
- `Customer_Name (`*`text`*`)`
- `Customer_Address (`*`text`*`)`

Capture specification
The capture specification defines the place in CICS where a particular event can be captured. Use capture points, filter predicates, and capture data to define the place where an event can be captured, the conditions that an event can be captured, and the data to be captured.

An event specification contains one or more capture specifications which are created by using the event binding editor. Each capture specification defines the following:

- Capture Point

  Defines the place in CICS where a particular event can be captured; for example, when an EXEC CICS READ FILE command is run.

- Filter predicates

  Define more precisely the conditions under which the event can be captured. Filters can be either:

  Context
  A filter relating to the context of the event, for example when the current transaction ID has a particular value.

  Event Option
  A filter relating to a particular value for a capture point option, for example, when the file has a specific name.

  Application data
  A filter relating to a particular value for piece of application data, for example when a field within a record read has a particular value.

  All predicates must be true for an event to be captured.

- Capture data

  Defines the data to be captured when the event occurs to satisfy the emitted business information items in the event specification.

Each event specification contains at least one capture specification. The specification describes exactly where in the program the event and its data are to be captured. The same event can occur at multiple points in the code; for example, because of conditional code branches or because the same event can be raised by multiple programs. So an event specification can contain multiple capture specifications.

Capture specifications support capture points for the following types of business event:

- Application event
- System event

Application events
An application event is a type of business event that results from application program activity and contains application data.

You can specify that an event is emitted when your application issues any of the event enabled EXEC CICS API commands or when an application program is initiated. The event enabled EXEC CICS commands are shown in Table 1 on page 7. A limited set of CICS modules have also been event enabled so that you can capture events from, for example: file and temporary storage commands coming from Atom support; EXEC CICS LINK commands issued by the CICS-WebSphere® MQ bridge programs; commands from the CICS samples and the CECI transaction.

The capture specification defines the criteria for event emission by using filters, such as the transaction ID, program name, or EXEC CICS command option value. This capture specification can be further refined by filtering on the application data associated with the command. For example, you can specify that the COMMAREA field containing an order value must be greater than 10,000. The capture specification also defines the location of the data to be captured.

The location of the event in the application logic depends on how it is specified. If a CICS application contains two instances of the same EXEC CICS API command, and the filter specification does not distinguish between the two commands, an event is emitted when both commands are issued. If the

same EXEC CICS API command occurs in two applications and there is no filter on the transaction ID or program name, an event is emitted when both applications run.

Capture points

You can capture events at program initiation (PGMINIT) and the following EXEC CICS API commands. Capture points that can be specified are shown in the following table:

Table 1. Application event capture points

Capture Point	Primary Predicate	Filter Predicate		Capture Data	
		Context	Event Option	Event option	Application Data
CONVERSE	None	Transaction Id Current Program User ID Response Code EIBAID EIBPOSN	None	FROM INTO-SET	None
DELETE FILE	FILE	Transaction Id Current Program User ID Response Code	FILE	FILE	RIDFLD
DELETEQ TD	QUEUE	Transaction Id Current Program User ID Response Code	QUEUE	QUEUE	None
DELETEQ TS	QNAME	Transaction Id Current Program User ID Response Code	QNAME	QNAME	None
INVOKE SERVICE	SERVICE	Transaction Id Current Program User ID Response Code	SERVICE OPERATION URI CHANNEL URIMAP	SERVICE OPERATION URI CHANNEL URIMAP	CHANNEL SCOPE
LINK PROGRAM	PROGRAM	Transaction Id Current Program User ID Response Code	PROGRAM CHANNEL	PROGRAM CHANNEL	COMMAREA CHANNEL
PROGRAM INIT	PROGRAM	Transaction Id User ID Response Code	PROGRAM CHANNEL	None	COMMAREA CHANNEL
PUT CONTAINER	CONTAINER	Transaction Id Current Program User ID Response Code	CONTAINER CHANNEL	CONTAINER CHANNEL	FROM

Table 1. Application event capture points (continued)

Capture Point	Primary Predicate	Filter Predicate		Capture Data	
		Context	Event Option	Event option	Application Data
READ	FILE	Transaction Id Current Program User ID Response Code	FILE UPDATE	FILE	RIDFLD INTO-SET
READNEXT	FILE	Transaction Id Current Program User ID Response Code	FILE UPDATE	FILE	RIDFLD INTO-SET
READPREV	FILE	Transaction Id Current Program User ID Response Code	FILE UPDATE	FILE	RIDFLD INTO-SET
READQ TD	QUEUE	Transaction Id Current Program User ID Response Code	QUEUE	QUEUE	INTO-SET
READQ TS	QNAME	Transaction Id Current Program User ID Response Code	QNAME	QNAME	INTO-SET
RECEIVE	None	Transaction Id Current Program User ID Response Code EIBAID EIBPOSN	NONE	NONE	INTO-SET
RECEIVE MAP	MAP	Transaction Id Current Program User ID Response Code EIBAID EIBPOSN	MAP MAPSET	MAP MAPSET	INTO-SET
RETRIEVE	None	Transaction Id Current Program User ID Response Code	NONE	NONE	INTO-SET
RETURN	None	Transaction Id Current Program User ID Response Code	TRANSID CHANNEL	TRANSID CHANNEL	COMMAREA CHANNEL

Table 1. Application event capture points (continued)

Capture Point	Primary Predicate	Filter Predicate		Capture Data	
		Context	Event Option	Event option	Application Data
REWRITE	FILE	Transaction Id Current Program User ID Response Code	FILE	FILE	FROM
SEND	None	Transaction Id Current Program User ID Response Code	NONE	NONE	FROM
SEND MAP	MAP	Transaction Id Current Program User ID Response Code	MAP MAPSET ALARM	MAP MAPSET	FROM
SEND TEXT	None	Transaction Id Current Program User ID Response Code	ALARM	NONE	FROM
SIGNAL EVENT [1]	EVENT	Transaction Id Current Program User ID Response Code	EVENT FROMCHANNEL	EVENT FROMCHANNEL	FROM FROMCHANNEL
START	TRANSID	Transaction Id Current Program User ID Response Code	TRANSID CHANNEL	TRANSID CHANNEL	FROM CHANNEL
WEB READ *FORMFIELD* HTTPHEADER QUERYPARM	None	Transaction Id Current Program User ID Response Code	NONE	NONE	FORMFIELD INTO-SET VALUE
WEB READNEXT *FORMFIELD* HTTPHEADER QUERYPARM	None	Transaction Id Current Program User ID Response Code	NONE	NONE	FORMFIELD VALUE
WRITE FILE	FILE	Transaction Id Current Program User ID Response Code	FILE	FILE	RIDFLD FROM

Table 1. Application event capture points (continued)

Capture Point	Primary Predicate	Filter Predicate		Capture Data	
		Context	Event Option	Event option	Application Data
WRITE OPERATOR	None	Transaction Id Current Program User ID Response Code	NONE	NONE	TEXT [2]
WRITEQ TD	QUEUE	Transaction Id Current Program User ID Response Code	QUEUE	QUEUE	FROM
WRITEQ TS	QNAME	Transaction Id Current Program User ID Response Code	QNAME	QNAME	FROM
XCTL	PROGRAM	Transaction Id Current Program User ID Response Code	PROGRAM CHANNEL	PROGRAM CHANNEL	COMMAREA CHANNEL

All capture points define the same context capture items, PROGRAM, TRANSID, and USERID.

Note:

1. The sole purpose of the SIGNAL EVENT API command is to provide data for an event. An event is only emitted if there is a matching capture specification enabled in the CICS system.
2. A length of 0 when specifying application data in the information sources tab means capture up to the end of the data area or container. This value is useful when you want to emit the contents of an EXEC CICS WRITE OPERATOR. For more information, see Information Sources tab in the CICS Explorer product documentation.

For more information about the capture points you can select, see Capture Point tab in the CICS Explorer product documentation and Information Sources tab in the CICS Explorer product documentation.

System events

A system event is a type of business event that results from system activity and contains system data. System events can include resource state changes, thresholds being crossed, or unusual system states or actions. Use system events to help you understand changes in the state of your system resources or system health.

Note:

CICS TS 5.4 introduces support for system rules in CICS policies. Policy system rules provide equivalent or enhanced capability to system events and supersede system events. For more information, see Policy system rules.

Support for system events is deprecated; no further enhancements will be delivered in any future CICS releases. If you use system events, you can continue to do so in this release, and the following information provides details about the supported system events. However, support for system events might be withdrawn in a future release of CICS, so consider migrating to use policy system rules at the earliest opportunity.

You can be alerted to certain CICS system conditions by capturing events for those conditions. Receiving a notification for any changes to the state of system resources avoids the need to poll for changes after they happen; it also means that you can quickly respond to these system events.

Event processing supports the following system events:

- DB2CONN connection status
- FILE enable or disable status
- FILE open or close status
- MESSAGE
- TASK threshold
- TRANCLASS TASK threshold
- Unhandled transaction abend

Capture points

The following table shows the capture points that are supported for system events.

Table 2. System event capture points

Capture point	Primary predicate	Filter Predicate Context	Filter Predicate Event Option	Capture Data Event Option	Description
DB2 CONNECTION STATUS	None	Transaction ID User ID	FROM_CONNECTST TO_CONNECTST	DB2ID DB2GROUPID DB2RELEASE FROM_CONNECTST TO_CONNECTST	You can capture an event whenever a DB2CONN connection status changes.[1]
FILE ENABLE STATUS	FILE	Transaction ID User ID	FILE FROM_ENABLESTATUS TO_ENABLESTATUS OPENSTATUS	FILE DSNAME FROM_ENABLESTATUS TO_ENABLESTATUS OPENSTATUS	You can capture an event whenever a file ENABLESTATUS changes.[1]
FILE OPEN STATUS	FILE	Transaction ID User ID	FILE FROM_OPENSTATUS TO_OPENSTATUS	FILE DSNAME FROM_OPENSTATUS TO_OPENSTATUS ENABLESTATUS	You can capture an event whenever a file OPENSTATUS changes.[1]
MESSAGE	MESSAGE_ID	Transaction ID User ID	MESSAGE_ID INSERT1 to INSERT30[2]	MESSAGE_ID INSERT1 to INSERT30 MESSAGE_TEXT	You can capture an event whenever CICS emits a DFHxxnnnn [3] message or CICSPlex® SM emits a EYUxxnnnn message.
TASK THRESHOLD	None	None	PERCENT_MAXTASKS	FROM_TASKS TO_TASKS MAXTASKS PERCENT_MAXTASKS	You can capture an event whenever a task threshold is crossed. The threshold is chosen from predefined lists.[6]
TRANCLASS TASK THRESHOLD	TRANCLASS	None	TRANCLASS PERCENT_MAXACTIVE	TRANCLASS FROM_ACTIVE TO_ACTIVE MAXACTIVE PERCENT_MAXACTIVE	You can capture an event whenever a TRANCLASS task threshold is crossed. The threshold is chosen from predefined lists.[6]

Table 2. System event capture points (continued)

Capture point	Primary predicate	Filter Predicate Context	Filter Predicate Event Option	Capture Data Event Option	Description
TRANSACTION ABEND (Unhandled)	TRANSACTION	User ID	TRANSACTION ABCODE	TRANSACTION ABCODE	You can capture an event whenever a transaction encounters any unhandled abend.

Notes:

1. The change can occur either through explicit operator actions, EXEC CICS SET commands, or implicitly as a result of CICS internal processing.
2. You can choose up to 10 message insert filters. Ensure that you use an available insert, because the CICS event binding editor does not prevent you from defining a filter on an unavailable insert and does not flag the error. Instead, the result is a runtime exception trace, the predicate evaluates as false, and no event is emitted. For example, message DFHFC0200 has 7 inserts. If you define a filter on INSERT 8 through 30, no event is emitted. Message inserts are shown in the individual message topics (see CICS messages).
3. You cannot event enable any of the following messages:
   - Any CICS initialization messages issued before event processing starts. Event processing is started just before phase 2 initialization PLT programs are run.
   - Any CICS termination messages issued after event processing stops. Event processing is stopped after all shutdown PLT programs have run.
   - Any messages sent to a CICS user, for example, messages issued by CICS supplied transactions such as CEMT and CEDA.
   - Any messages issued by the EC or EP components, that is, all DFHECnnnn and DFHEPnnnn messages.
4. A MESSAGE system event is emitted regardless of whether or not a message is suppressed by either an XMEOUT global user exit program or the system initialization parameter **MSGLVL=0**.
5. No transaction abend system events are emitted for any transaction abends originating in any task running an event processing adapter program.
6. Task threshold values are chosen from predefined lists as follows:
   - 60%, 70%, 80%, 90%, or 100% when the 'Goes Higher Than" operator is used.
   - 50%, 60%, 70%, 80%, or 90%, when the 'Goes Lower Than" operator is used.

For more information about the capture points you can select, see Capture Point tab in the CICS Explorer product documentation and Information Sources tab in the CICS Explorer product documentation.

Task thresholds

You can use task threshold events to monitor both system and transaction class definition (TRANCLASS) task load by specifying that an event is to be emitted when the active task count crosses a threshold value.

Task thresholds are expressed as a percentage of the **MXT** system initialization parameter or the **MAXACTIVE** value of the TRANCLASS resource. For performance reasons, they are predefined. The possible task threshold values you can select are:

- 60%, 70%, 80%, 90%, and 100% when the "Goes Higher Than" operator is used.
- 50%, 60%, 70%, 80%, and 90% when the "Goes Lower Than" operator is used.

You can filter on:

- A threshold to indicate how close the system is to the **MXT** value. You can define more than one event to indicate various degrees of health as the threshold gets closer to the limit you set for the **MXT** system initialization parameter.

- A TRANCLASS resource and a threshold to indicate how close the TRANCLASS resource is to its **MAXACTIVE** value. You can define more than one event to indicate various degrees of health as the number of attached tasks gets closer to the **MAXACTIVE** limit you set for the TRANCLASS resource.

To avoid large numbers of events being emitted, events are emitted only when the number of active tasks crosses a new threshold boundary. For example, events are emitted during transaction attach when the number of active tasks exceeds a threshold and the previous "goes higher than" event was when the next lowest threshold was crossed. Events are emitted during transaction detach when the number of active tasks falls below a threshold and previously "goes lower than" event was when the next highest threshold was crossed. Figure 3 on page 13 shows such examples.

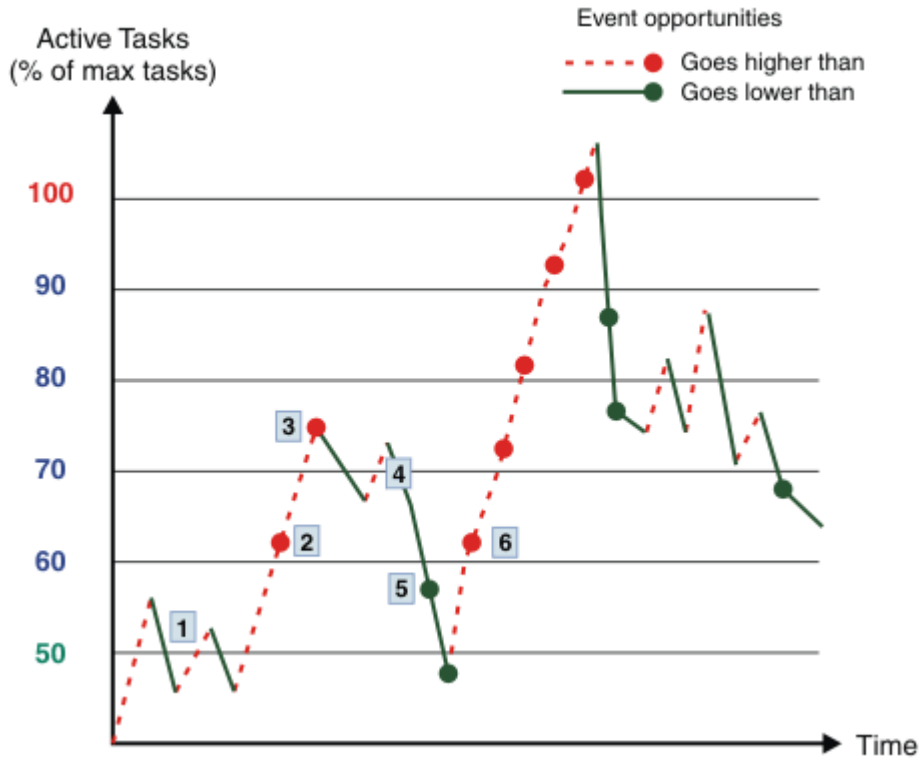


Figure 3. Event emission opportunities

1 No events are emitted when the number of active tasks exceeds the 50% threshold. Events are emitted during transaction attach only when the number of active tasks crosses one of the 60%, 70%, 80%, 90%, or 100% thresholds. No events are emitted when the number of active tasks falls below the 50% threshold, because the number of active tasks has not been above the 60% threshold since the last "goes below 50%" event.

2 An event is emitted because the number of active tasks exceeds the 60% threshold for the first time since the previous "goes above 50%" event.

3 An event is emitted because the number of active tasks exceeds the 70% threshold for the first time since the previous "goes above 60%" event.

4 No events are emitted when the number of active tasks oscillates around the 70% threshold because the number of active tasks does not cross a threshold boundary to either exceed the 80% threshold or drop below the 60% threshold.

5 An event is emitted because the number of active tasks drops below the 60% threshold for the first time since the previous "goes below 70%" event.

6 A series of events are emitted when the number of active tasks progressively exceeds the 60%, 70%, 80%, 90%, and 100% thresholds.

Considerations

Event emission is not enabled for a TRANCLASS when its MAXACTIVE value is set to less than 10.

TRANCLASS threshold events are not emitted for those transactions defined as not having a TRANCLASS, that is, those defined with TRANCLASS(DFHTCL00).

If you require an event to be emitted when the system or TRANCLASS crosses the 100% threshold, and you need that event emitted as quickly as possible, consider the dispatch characteristics of the EP adapter.

- For the system 100% threshold, ensure that the EP adapter will be linked to. It is possible that an attached EP adapter task would be queued until the MAXTASK condition is cleared.
- For the TRANCLASS 100% threshold, ensure that the TRANCLASS that causes the event is not used for the EP adapter.

Event processing (EP) adapter specification

An EP adapter specification is an XML definition that defines how events that use the specification are formatted and emitted.

You create an EP adapter in a CICS bundle project using either the Event binding editor or the EP adapter configuration editor. EP adapters are included in bundles; bundles are the units for deploying, enabling, and disabling EP adapters. For more information about bundles, see BUNDLE resources.

An EP adapter specification defines the type of adapter to be used to process the emitted events, associated adapter configuration information, event dispatcher information, and options to configure event emission.

An EP adapter specification can be either:

- Embedded

  An embedded EP adapter is defined by using the adapter tab of the event binding editor and is saved by the editor in an XML event binding file in a CICS bundle project.

- Separate

  A separate EP adapter is defined by using the EP adapter configuration editor and is saved by the editor in a CICS bundle project as an XML EP adapter file that has the suffix `.epadapter`.

When you use a separate EP adapter specification you can easily share it between event bindings, resulting in only one EPADAPTER resource to manage. When you specify an embedded EP adapter CICS creates an EPADAPTER resource for you with the same name as the event binding. For more information, see CICS Explorer product documentation.

You can deploy the CICS bundle project to a CICS system. You can define, install, enable, disable, and uninstall a deployed CICS bundle in a CICS system. In addition, you can enable and disable individual deployed EPADAPTER resources.

To deploy to a CICS system, an EP adapter specification must conform to the EP adapter schema files installed with CICS in `/usr/lpp/cicsts/cicsts56/schemas/eventprocessing/eventbinding`:

- `CicsEPAdapter.xsd`
- `CicsEventDispatching.xsd`

For more information about schemas, see Event processing schema versions in the CICS Explorer product documentation.

Event processing (EP) adapter set specification

An EP adapter set specification is an XML definition that contains multiple EP adapter names. By using an EP adapter set, you can format and emit events to multiple EP adapters.

You create an EP adapter set in a CICS bundle project using the CICS EP adapter set configuration editor. EP adapter sets are included in bundles; bundles are the units for deploying, enabling, and disabling CICS EP adapter sets. For more information about bundles, see BUNDLE resources.

An EP adapter set specification defines the names of adapters to be used, is defined by using the EP adapter set configuration editor, and is saved by the editor in a CICS bundle project as an XML EP adapter set file that has the suffix `.epadapterset`.

You can easily share EP adapter sets between event bindings, resulting in only one EPADAPTERSET resource to manage.

You can deploy the CICS bundle project to a CICS system. You can define, install, enable, disable, and uninstall a deployed CICS bundle in a CICS system. In addition, you can enable and disable individual deployed EPADAPTERSET resources.

To deploy to a CICS system, an EP adapter set specification must conform to the EP adapter set schema file installed with CICS in `/usr/lpp/cicsts/cicsts56/schemas/eventprocessing/` :

- `CicsEPAdapterSet.xsd`

For more information about schemas, see Event processing schema versions in the CICS Explorer product documentation.

How CICS event processing works

You can configure CICS to detect and emit application events or system events. When a specified event occurs, CICS collects data and triggers a synchronous or asynchronous process, through an EP adapter.

CICS can be configured to detect and emit both application and system events from the later stages of CICS Initialization through until the end of the first stage of CICS shutdown. Event processing is started before the second phase of CICS initialization PLT processing begins, and quiesced after first phase of CICS shutdown PLT processing completes. However, events triggered by any new tasks which are started by a CICS shutdown PLT program are not guaranteed to be detected.

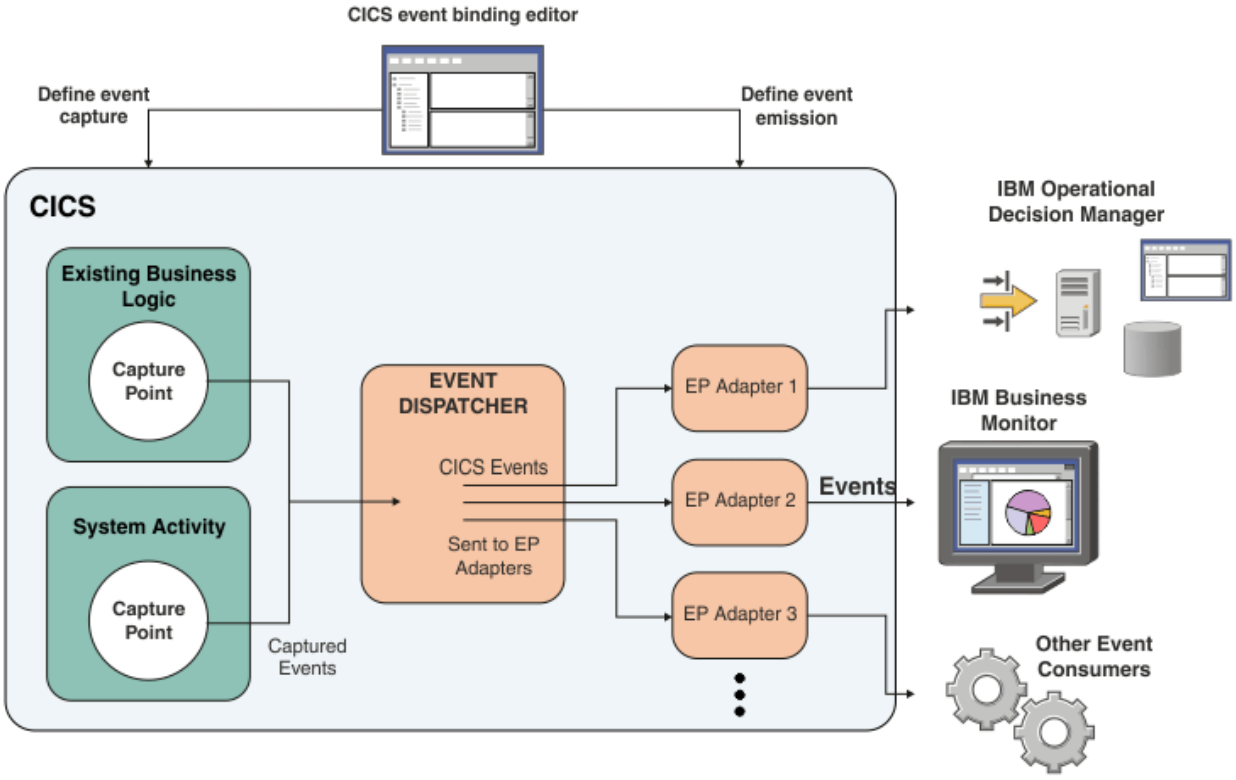


Figure 4. Event processing high-level architecture

Note: IBM Business Monitor, formerly known as WebSphere Business Monitor.

1. You specify when to capture and emit events by using the event binding editor.

- An event binding is created in a CICS bundle project
- The criteria for capturing an event and the data to be included within the event are known as capture specifications and are saved in event bindings. Each capture specification relates to a particular capture point, for example following the EXEC CICS READ FILE API call or the point at which a file enable status is changed.
- The event emission options including the required format and destination are specified in an EP adapter.

2. You export the bundle to the z/OS UNIX file system and install it into CICS by using a BUNDLE resource.
   - CICS creates EVENTBINDING and EPADAPTER resources from the BUNDLE.
3. CICS checks for active capture specifications whenever an event capture point is run.
   - If capture specifications are active at the capture point, CICS checks the filter criteria and captures the specified data for any that evaluate true.
4. Captured events are passed to the event dispatcher for emission.
5. The event dispatcher passes each event to the EP adapter specified for the event binding.
6. The EP adapter formats the event and emits it to a destination according to its configuration options.

How the event is processed after it has been captured depends on the Emission mode and Transactional mode of the EP adapter used to emit it.

Transactional mode

Transactional mode can be used to control whether event emission is dependent on the success of the capturing unit of work.

- Non-Transactional: Events are available for emission as soon as they are captured.
  - Use non-transactional when you want the event emitted regardless of whether the capturing unit of work completes successfully. For example, an event for *Order Attempted* should be emitted non-transactional because you would want it to be emitted even if the unit of work did not complete successfully and the order was not placed.
  - Events can be emitted even if the capturing unit of work backs out, therefore it is possible to get events even if something was backed out or never committed, and also duplicate events as work is tried again.
  - A benefit is that you do not have to wait for sync point before the event can be emitted, this transactional mode can be useful if capturing events from long running tasks, as events are emitted immediately.
- Transactional: Events are emitted if and only if the unit of work from which they were captured completes successfully.
  - Use transactional only when you want the event to be emitted if a task completes successfully. For example, the *Order placed* event should be emitted only if the order was completed successfully.
  - As events are not emitted for units of work that are backed out or not committed, duplicate events are avoided if work is tried again.
  - A disadvantage is that if capturing the events from a task which waits a long time between sync points the events will be emitted some time after they actually occur.
  - Transactional emission is not available for system capture points. System events are typically not transactional and you would not want to wait for a system unit of work to complete before emitting a system event; you would want to emit the system event as soon as possible. You will receive a DFHEC1023 error message in the CICS message log after the emission of the first system event for any event binding which references a transactional EP adapter.

Emission mode

Emission mode can be used to control whether an event is emitted asynchronously, or as part of the capturing unit of work.

- Asynchronous: When an event has been captured it is added to a dispatcher queue. Control is returned to the unit of work from which it was captured and the event is formatted and emitted asynchronously.
  - Use Asynchronous emission wherever possible because it has less impact on the response time of the capturing application and does not change its behavior. Specifically it should be used when the capturing unit of work can be considered complete even if the event fails to be emitted. For example, an *Order over $1,000* event that is used for statistics, monitoring, or targeting promotional offers should be emitted asynchronously as the order is complete and valid with or without the event.
  - For asynchronous emission the recoverability of the transport is not a concern because the correct transactional behavior is achieved by CICS either emitting the event immediately or waiting for sync point.
  - Event emission failures do not change the behavior of the application.
  - A benefit is that asynchronous emission has a low performance impact on capturing application.
  - A disadvantage is the potential to lose events that failed to emit, for example, due to network outages or across a CICS restart. Applications from which the events were captured can complete successfully regardless of whether the event was emitted, therefore the emission of events with emission mode asynchronous is not assured.
- Synchronous: When an event has been captured it is formatted and emitted as part of the unit of work from which it was captured. If the event fails to be emitted the capturing transaction will be backed out at sync point with an ASP7 abend code.
  - Use synchronous emission when the capturing unit of work is considered incomplete if the event failed to be emitted. For example, when an *Order placed* event is used to trigger the next step in an event driven order handling process.
  - The behavior of the application can be affected by event emission failures. For example, when emitting an *order placed* event, orders cannot be processed if your event consumer in unavailable.
  - A benefit is that events are not lost if they fail to be emitted because the actions which caused them are backed out and therefore never happened. Events defined with an emission mode of synchronous are assured and can be used as part of event driven processes.
  - A disadvantage is that there is a greater performance impact on the application. Consider the impact to response times as formatting an emission of the event now takes place on the applications thread.
  - Synchronous event emission is not available for system event capture points. Since system events are typically used for statistics and monitoring they do not typically have the requirement for synchronous emission. A DFHEC1024 error message is received in the CICS message log after the emission of the first system event for any event binding which references an EP adapter that is defined with synchronous emission mode.

How CICS assures event emission

You can assure emission of an event by using an EP adapter with synchronous emission mode and the appropriate transaction mode. With synchronous emission, event formatting and emission processing is completed synchronously within the unit of work of the capturing transaction. The unit of work completes successfully only if the event is emitted.The EP adapter is linked from the application task; that is, the event is not queued for asynchronous processing by an EP dispatcher thread or separate EP adapter task.

Synchronous event emission is not supported for system capture points therefore emission of system events cannot be assured.

Synchronous events can be transactional or non-transaction but the recoverability of the transport must be set up correctly in each case.

Synchronous non-transactional events:
The EP adapter must emit events to its transport in an unrecoverable manner so that the events are not backed out if the unit of work fails.

Synchronous transactional events:
The EP adapter must emit events to its transport in a recoverable manner so that transactional events are backed out if the unit of work is backed out.

Not all EP adapters can support synchronous emission with all combination's of **TRANSMODE**. For more information, see Event processing adapters.

When an event fails to be emitted, the EP adapter provides information about the event and why it was not emitted, increments relevant event statistics, and causes the unit of work for the capturing transaction to be backed out.

Considerations for planning

Understanding the way that synchronous event emission works helps you to understand how to make the best use of this capability. Some things to consider when you use synchronous event emission include: security, performance, transport, and effects on applications.

The event-capturing transaction must have write authority to the event emission transport (for example, the IBM MQ queue for the WebSphere MQ EP adapter) for synchronous emissions; the EP dispatcher or adapter task that emits events needs write authority for asynchronous emission.

Synchronous, transactional event emission is recoverable. When you use the CICS WebSphere MQ EP adapter, events are put on the IBM MQ event queue under sync point; therefore, you might have to review your IBM MQ log data set space allocation. When you use the CICS TSQ EP adapter, this adapter increases the use of your recoverable TS queue so you might have to review your CICS log stream size and attributes. Using synchronous transactional events with a task that runs for a long time without taking a sync point can cause the log to overflow.

When using synchronous emission, a custom EP adapter must honor the EPAP_RECOVER flag in the DFHEP.ADAPTPARM container. For more information, see Custom EP adapter.

Assuring your event emission provides the opportunity to build business critical event-based applications, and extend existing applications in a reliable way. The trade-off is that the synchronous processing that is essential to assuring that events are emitted might have an impact on your application response time. A judicious use of synchronous event emission minimizes the application impact. See Improving event processing performance for more information about performance considerations when assuring event emission.

A single unit of work might cause many events to be emitted, where some events are transactional and some events are non-transactional. If a capturing transaction fails to emit a synchronous event, the unit of work is backed out with any transactional events that it captures. Non-transactional events might still be emitted.

The EP adapter, its resources (such as an IBM MQ queue), and the event consumer must be configured with enough capacity to process the expected peak volume of events to be emitted in order to prevent the capturing transaction from failing.

To help you decide where to use synchronous emission, here are some considerations for asynchronous and synchronous emission for comparison:

Consider the following characteristics when you are using asynchronous event emission:

- The order in which events are emitted from a CICS program can be different from the order in which they were captured.
- The order in which events are emitted can vary from one run to another of an event-enabled CICS program.
- Event emission is not assured. Events might be lost if CICS ends abnormally with events in progress. In the unlikely event of a CICS system failure after the capture of an event but before its emission, the

event might not be emitted at all, regardless of whether the EP adapter in the event binding specifies that events are transactional.

- The cost of formatting and emitting the event is offloaded to an event processing thread or to a separate transaction.
- There is a minimal impact on the capturing transaction.
- No changes are required to application code.

Consider the following characteristics when you are using synchronous event emission:

- Events are emitted from a CICS program in the order in which they were captured.
- Event emission is assured when the capturing transaction completes successfully.
- Events can be considered as application extensions for business-critical data.
- When synchronous emission is used with the WebSphere MQ EP adapter, event delivery can be assured.
- The cost of formatting and emitting the event is added to the application thread.
- There is a greater impact on the capturing transaction; for an event emitted on an application thread, an event emission failure causes the application to back out. Consider the capturing transaction configuration and the impact on overall system resource usage.

Understanding the events that are enabled in your CICS system

You can use CICS Explorer to browse event bindings and capture specifications. For example, the **Event Bindings** view shows the bundles that are installed in different regions and whether the bundles are enabled.

Filters can be used in capture specifications to restrict when events are emitted.

As a system programmer, you might want to understand more about any application context or primary predicate filters that are set for a given capture specification. The **INQUIRE CAPTURESPEC** command has options that return detailed information about the primary predicate and the application context or system context filters for the capture specification. For example, you can use the **INQUIRE CAPTURESPEC** command to discover all of the capture specifications that include a predicate on a certain program. The **INQUIRE CAPDATAPRED**, **INQUIRE CAPINFOSRCE**, and **INQUIRE CAPTOPTPRED** commands have options that return further detailed information about all the predicates and information sources defined for the same capture specification.

Event processing terminology

Many terms that are used in CICS event processing are in the glossary. Additional terms are described in this topic.

For the following terms and definitions that are used in CICS event processing, see CICS glossary:

- application context
- application context data
- application data
- application event
- assured event emission
- business event
- capture data
- capture point
- capture specification
- CICS bundle
- context data

- emit
- emitted business information
- event binding
- event capture
- event dispatcher
- event option
- event processing adapter (EP adapter)
- event processing adapter configuration
- event processing adapter set
- event specification
- filter
- predicate
- primary predicate
- system context
- system event

Definitions for additional terms that are used in CICS event processing are as follows:

CICS container-based event format (CCE format)

The format used, for example, to start the processing in a CICS transaction as a result of an event.

CICS flattened event format (CFE format)

A format used to write events to a temporary storage queue for testing or to an IBM MQ queue, from which they can be read from by a program running in or outside CICS.

common base event format

A format that is used to emit events for consumption by IBM Business Monitor.

common base event REST format

A format used to transfer events to the IBM Business Monitor REST interface.

deferred filter request

Event filter and capture that is performed under the CEPF system task rather than the event-generating task, to avoid suspending critical system tasks.

Decision Server Insights Event (DSIE) format

An XML format used to emit events for consumption by the Decision Server Insights component of IBM Operational Decision Manager.

event binding file

An event binding that is expressed as an XML document saved as a text file with a `.evbind` file extension.

event data

Data included with the emitted event (the emitted business information).

WebSphere Business Events format

A format used to emit events for consumption by Decision Server Events component of IBM Operational Decision Manager.

Chapter 2. Setting up event processing

Before you can use event processing, your CICS system must be correctly configured. You can stop and start event processing in CICS using a number of methods, including using IBM CICS Explorer.

About this task

You might want to stop event processing for an upgrade or system maintenance, and then start event processing again. This task explains the process usingCICS Explorer.

Use the LOCALCCSID system initialization parameter to specify the coded character set identifier for your local CICS region. This is the default code page for application programs. The default setting for LOCALCCSID is the EBCDIC code page 037. CICS translates the data content of event binding files into this code page before using it for event capture filtering operations, and assumes that any captured character data is in this code page.

Event processing is enabled by default when a START=INITIAL or START=COLD parameter is used during the startup of your CICS systems.

When a START=WARM or START=EMERGENCY parameter is used, the settings from the previous run of CICS are used.

Note: Do not change the status of event processing (that is, set to start, drain, or stop) while a unit of work that captures synchronous transactional events is in progress because you might cause the events to be backed out and the transaction to end abnormally.

Procedure

Use this task to stop or start event processing.

1. Using the CICSplex Explorer view in CICS Explorer, click the CICSplex or CICS region to select the CICSplex or CICS region on which you want to stop event processing.
2. Using the CICS Explorer toolbar, click **Operations** > **Event Processing**.

   The status of event processing is displayed.
3. Right-click to select the region on which you want to stop event processing and click **Stop**. Then click **OK**.

   Note: The status does not update until you click the **refresh** icon at which point the STOPPED status is displayed in the **Event Processing Status** column, indicating that event processing is stopped.
4. To restart event processing, right-click to select the region for which you want to start event processing and click **Start**. Then click **OK**.

   Note: The status does not update until you click the **refresh** icon at which point the STARTED status is displayed, indicating that event processing is started.

Results

When event processing is STOPPED, event capture is stopped immediately. All events on the dispatcher queue are deleted.

When event processing is STARTED, for in flight transactions, the capture of nontransactional events starts immediately and the capture of transactional events starts at the next sync point.

What to do next

You must enable an event binding and an EP adapter before you can emit any events for processing by an event consumer. For more information, see Enabling an event binding in the CICS Explorer product documentation and Enabling an EP adapter in the CICS Explorer product documentation.

If you are using the WebSphere MQ EP adapter to format and process your events, you must set up the CICS-WebSphere MQ adapter to provide a connection between the CICS region and IBM MQ on z/OS. For instructions to do this, see Setting up the CICS-MQ adapter.

Chapter 3. Improving event processing performance

CICS Event Processing (EP) consists of three main components: capturing an event, dispatching the EP adapter, and emitting the event by running the EP adapter. For each component, different factors influence performance.

Factors when capturing an event

For CICS performance associated with running EP, consider the following factors:

- Processor usage is negligible with EP started and no event binding file installed.
- Processor usage is negligible with EP started and event binding files installed, but when the capture point is not defined in any <locationFilter> (the capture point and application command predicates) element in any of the Event Capture Specifications.
- For primary predicate matching, use of the attribute **`filterOperator="EQ"`** ('Equals' specified in the filtering predicate) can result in improved performance, compared to other **`filterOperator`** values. An optimization is used when **`filterOperator="EQ"`** is specified.

  Note: The CICS event binding editor uses an asterisk (*) to show the primary predicate for the capture point selected.
- If you are using <dataCapture> elements (**Emitted Business Information** items mapped in the **Information Sources** part of the capture specification) to capture specific parts of the data associated with a capture point, keep the number of <dataCapture> elements reasonably low. Otherwise, performance can be adversely affected, because a separate container is created for each data capture component. For example, if bytes 0 - 5, 10 - 15, 20 - 25, 30 - 35 of a record associated with an **`EXEC CICS WRITE`** command are to be captured, it is more efficient to define a single <dataCapture> element to capture bytes 0 - 35 than to define four <dataCapture> elements for each of the separate data areas.
- Where possible, do not include unnecessary filter predicates. For example, it is often not necessary to filter on both the transaction ID and the current program name.
- Put the filter predicates that exclude the most unwanted events before the filter predicates that exclude fewer unwanted events.
- Where possible, do not use filter predicates on zoned decimal or packed decimal data fields that might contain unusable data.

CICS statistics in the EVENTBINDING GLOBAL STATISTICS category show the Total Event Filter operations. For more information, see EVENTBINDING statistics.

Factors when dispatching the EP adapter

EP runs multiple dispatcher tasks, which run on L8 TCBs, to service the emitting of events. These dispatcher tasks count toward the limit set by CICS for the number of L8 and L9 mode open TCBs. CICS sets this limit automatically using the formula `(2 * MXT value) + 32`. So that EP does not monopolize the L8 TCBs, the maximum number of L8 TCBs used for EP dispatchers is limited to one third of the open TCB limit. CICS statistics in the EVENTPROCESS STATISTICS category show the peak event capture queue and the peak dispatcher tasks so that you can monitor the use of L8 TCBs by EP. For more information, see EVENTPROCESS statistics.

Adapters can be either attached or linked depending on the adapters configuration. Attaching a task for each event increases processor usage; using the attach method means that the event dispatcher task does not wait for the event to be emitted to continue processing the event queue.

When a transaction ID or user ID is specified in the CICS event binding editor adapter section, the adapter is attached as a separate task. When a transaction ID or user ID is not specified, the adapter is linked to from the dispatcher task. However, the HTTP EP adapter is always attached.

Factors when emitting events by running the EP adapters

The cost of running the EP adapters depends on the EP adapter that is being used:

TSQ
The TSQ EP adapter imposes the least cost in terms of CPU consumed.

WebSphere MQ
Each event has an MQPUT1 call for the WebSphere MQ EP adapter.

Assured event emission: You need the capability and reliability of assured event emissions to build business-critical application extensions based on events. However, assuring event emission (using synchronous emission mode) transfers the cost of event emission from an asynchronous event processing task to the application thread. Therefore, assured event emission could have an adverse affect on application response time, similar to adding an MQPUT for the event to the application itself. Because of the effect on response time, consider carefully which events demand the added level of reliability that is achieved through synchronous emission mode. Asynchronous emission mode is the better choice when it is less important that an event is occasionally lost (between the time an event is captured but not yet emitted); for example, for mail applications when an event is infrequently lost. Specifying synchronous emission mode only where it is needed ensures that your business is making the best use of assured event emission with the least impact on performance.

HTTP
Each event has a `WEB OPEN`, `WEB CONVERSE`, and a `WEB CLOSE` call for the HTTP EP adapter. By default, CICS closes the client HTTP connection after the event is emitted, and a new connection is opened for the next event. CICS can keep client HTTP connections open after events are emitted, so that they can be reused for subsequent event emissions, and you can save the processor overhead for reopening the connections. To keep connections open, specify the SOCKETCLOSE attribute in the URIMAP resource that the HTTP EP adapter uses for the connection. Choose an appropriate expiry time for the connections based on your emission rate.

A CEPH task is attached for each event emitted using the HTTP adapter. The CEPH transaction has a **DTIMEOUT** value of 5 seconds so that the HTTP EP adapter tasks timeout if a connection cannot be established with the event server within 5 seconds. The CEPH task then emits the event to an HTTP 1.1 compliant server. CICS provides a PROFILE for the CEPH transaction called DFHECEPH. This profile has an **RTIMOUT** value of 5 seconds so that the HTTP EP adapter tasks timeout if the event server does not respond within 5 seconds. You can copy this profile and the CEPH transaction if you want to change them.

- If the emission rate is high compared to the HTTP 1.1 compliant server or network response time, these CEPH tasks can flood the CICS system resulting in the MXT limit being reached. You must copy the CEPH transaction and then assign your transaction to a transaction class to avoid the MXT limit being reached. The transaction class must have a **MAXACTIVE** value low enough to avoid reaching the MXT limit, and a **PURGETHRESH** value set to a non-zero value.
- Any CEPH tasks that are purged when the **PURGETHRESH** limit is reached do not emit their events. Any tasks purged when the **PURGETHRESH** limit is reached result in the following message being written to the CSMT transient data destination: `DFHAC2036 Transaction CEPH has failed with abend AKCC`.

Note: When you use the copied profile or transaction, you must change your adapter configuration in the Event binding editor to run the HTTP adapter using this new transaction.

Transaction start
The transaction start EP adapter starts a new CICS task. Therefore, processor usage increases overall because of starting the transaction that is driven as a result of the CICS event. Also, each data capture field is made available to a started transaction in a container. A large number of these CICS tasks can impose a significant increase in processor usage on the adapter.

CICS statistics in the EVENTPROCESS STATISTICS category show the number of the following events:

- Events dispatched to the WebSphere MQ EP adapter
- Events dispatched to the HTTP EP adapter

- Events dispatched to the Transaction EP adapter
- Events dispatched to the TSQ EP adapter
- Events dispatched to the Custom EP adapter

For more information, see EVENTPROCESS statistics.

CICS region storage for event processing

Event processing uses 64-bit (above-the-bar) storage in the CICS region. Your use of event processing therefore influences the value that you choose for the z/OS **MEMLIMIT** parameter that applies to the CICS region.

If you use the temporary storage queue (TSQ) adapter and select main temporary storage, this data is also in 64-bit storage.

For information about the **MEMLIMIT** value for CICS, and instructions to check the value of **MEMLIMIT** that currently applies to the CICS region, see Estimating, checking, and setting MEMLIMIT in Improving performance. For further information about **MEMLIMIT** in z/OS, see Limiting the use of private memory objects in the z/OS MVS Programming: Extended Addressability Guide.

Chapter 4. Scenario: Using Atom feeds and event processing to manage customer orders

CICS can expose files as Atom feeds so that external parties can read and update the contents of the files. Event processing can monitor any changes made to the files.

You can use Atom feeds to allow external parties to read and update files within your CICS system without granting the external party access to your CICS system. This scenario shows how you can use Atom feeds to allow a third-party company to complete order fulfilment. The status of the order fulfilment can be monitored using event processing.

Planning the solution

Review the topics in this section to understand what is covered in this scenario, the reasons why a business might want to follow the scenario, and to see an overview of the solution proposed by the scenario.

Assumptions

This scenario makes several assumptions about your system, such as the version of the products that you are using.

This scenario makes the following assumptions:

- You are using CICS TS for z/OS, Version 4.2 or later.
- You are using a version of Java™ that is supported by your CICS release.
- You have CICS Explorer installed. See Downloading and starting CICS Explorer in the CICS Explorer product documentation for more information.
- TCP/IP support is enabled for your CICS system. For more information, see Enabling TCP/IP in a CICS region.
- Your CICS system is correctly configured to access z/OS UNIX System Services by including an OMVS segment in the user profile of the CICS region user ID. For more information, see Authorizing access to z/OS UNIX System Services.
- Your CICS system is correctly configured for event processing. For more information, see Configuring your CICS system for event processing.
- Your CICS region has a TCPIPSERVICE resource definition for the port that you want the order processing application to make HTTP requests for the Atom collection. For more information, see Creating TCPIPSERVICE resource definitions for CICS web support.
- The new orders, fulfilled orders, and unfulfilled orders VSAM files exist in a CICS region with an associated copy book. The structure of the copy book must match the following structure:

```
 01 ORDER-DETAILS.
  10 ORDER-NUMBER     PIC X(5).
  10 ITEM-ID          PIC X(5).
  10 ITEM-DESC        PIC X(50).
  10 ITEM-QUANT       PIC S9(8) BINARY.
  10 ITEM-PRICE       PIC X(6).
  10 FULFILLED        PIC X.
    88 COMPLETED      VALUE 'Y'.
    88 INCOMPLETE     VALUE 'N'.
*71 BYTES IN LENGTH
*KEY IS 5 BYTES FROM OFFSET 0
```

Business overview

A company wants to use a third-party company for order fulfillment, but does not want to grant the third-party company access to their CICS systems to complete the fulfillment process.

Company A manufactures a range of products but wants to use a third-party company, Company B, to manage order fulfillment. Company A stores order data for all products in a file in a CICS region. To process orders, Company B must obtain the order data from this file. However, Company A does not want to grant Company B access to their CICS systems. Using Atom collections, Company A can expose order data to Company B without granting access to their CICS systems.

Using Atom collections, Company B can request order data from Company A. Company B must then process the orders. Depending on the levels of stock for the items ordered, Company B must notify Company A if the order has been fulfilled or is currently unfulfilled due to lack of stock.

Company A needs to be alerted to the notifications sent by Company B relating to the status of the orders. Company A can use event processing to monitor for order status notifications.

Technical solution

Atom collections can be used by Company A to expose order data without granting Company B access to their CICS system. Event processing can be used to monitor the status of the orders that Company B has processed.

Company A stores order data in a New Orders VSAM file. When an order for any product is made, the order is appended to the new orders file. Company A must expose the contents of new orders file as an Atom collection. By exposing the contents of the file as an Atom collection, Company B can make GET, POST, PUT, and DELETE requests to the Atom collection. For more information about Atom collections, see Atom documents. The Atom collection, therefore, enables Company B to access and update the order data. Company A must also create two other VSAM files for fulfilled orders and unfulfilled orders. These files must also be exposed as Atom collections so that Company B can notify Company A of status of each order by making a POST request to the appropriate file.

The processing application (OPA) used by Company B must be able to issue Atom collection requests to the Atom collections supplied by Company A. To obtain the order data, the OPA must make a GET request to the new orders Atom collection. As a result of this GET request, the new orders Atom collection supplies the OPA with the order data in the new orders file. The OPA determines if there is sufficient stock levels of the products ordered to fulfill the order.

If there are sufficient levels of stock then the order can be fulfilled. When the order has been processed the OPA notifies Company A by issuing a POST request to the fulfilled orders Atom collection. Company A can use event processing to monitor for POST request to this collection. When a POST request is made, an event is triggered to alert Company A that an order has been successfully processed.

If there is insufficient stock of the product to fulfill the order, the order is not processed. The OPA notifies Company A of an unprocessed order by issuing a POST request to the unfulfilled orders Atom collection. Company A can use event processing to monitor for POST request to this collection. When a POST request is made, an event is triggered to alert Company A that an order has not been processed due to a lack of stock.

After the OPA sends the appropriate POST request to either the fulfilled orders or unfulfilled orders Atom collections, the original order can be deleted from the new orders file. To do this, the OPA sends a DELETE request to the new orders Atom collection.

Figure 5 on page 29 shows a flow diagram of the order fulfillment process, as described preceding paragraphs, including the Atom collection requests and responses and the event processing triggers.

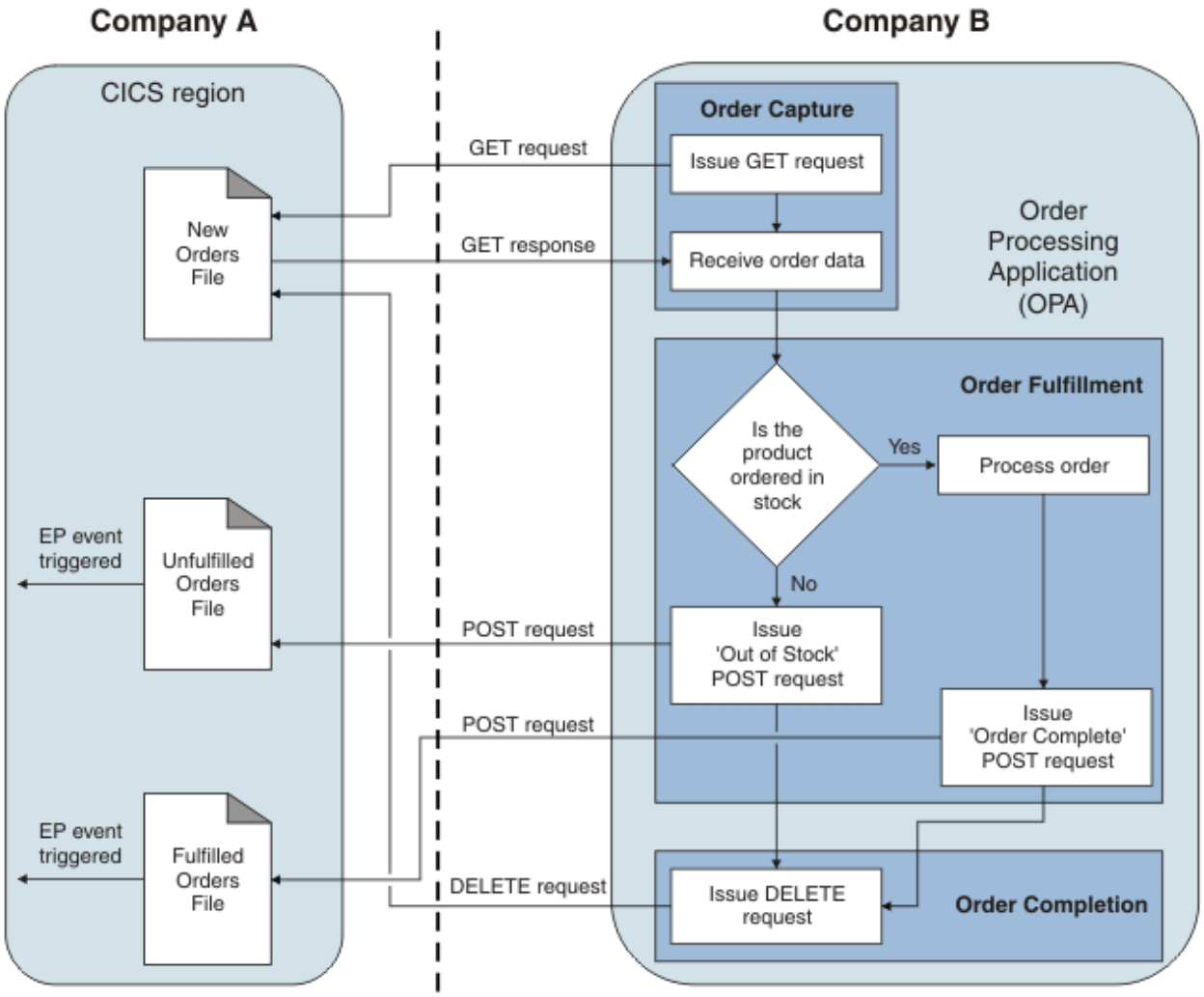


Figure 5. Flow diagram describing the Atom feed and event processing concepts for this scenario

Implementing the solution

Implementing the solution in this scenario involves creating a bundle, creating three Atom collections, monitoring two VSAM files using event processing, and the order processing application.

Before you begin

The starting point for this scenario assumes that the new orders, fulfilled orders, and unfulfilled orders VSAM files exist in a CICS region with an associated copy book.

It is assumed that your CICS system is correctly configured for web support. For more information, see Configuring CICS web support components. Your CICS region must have a TCPIPSERVICE resource definition for the port that you want the order processing application to make HTTP requests for the Atom collection. For more information, see Creating TCPIPSERVICE resource definitions for CICS web support.

It is also assumed that your CICS system is correctly configured for event processing. For more information, see Configuring your CICS system for event processing.

About this task

The following tasks describe how to expose the new orders, fulfilled orders, and unfulfilled orders VSAM files as Atom collections. Also described is how to configure your CICS region to use event processing to

monitor for the POST request to the fulfilled and unfulfilled VSAM files. The last step of the details how to create the order processing application.

Creating and installing the Atom collections resources

To expose the VSAM files as Atom collections you can use the CICS Explorer to create a bundle project containing the XML binding, schema, and three Atom configuration files, one configuration file for each VSAM file.

About this task

Complete the tasks listed in the following procedure to set up CICS definitions to expose the new orders, fulfilled orders, and unfulfilled orders VSAM files as Atom collections.

It is assumed that the new orders, fulfilled orders, and unfulfilled orders VSAM files exist in a CICS region with an associated copy book. For this scenario, the FILE resources associated with these files are called ORDERS, UNFULFIL, and FULFILL respectively. The copy book must have the file extension .cbl or .cpy with the following structure:

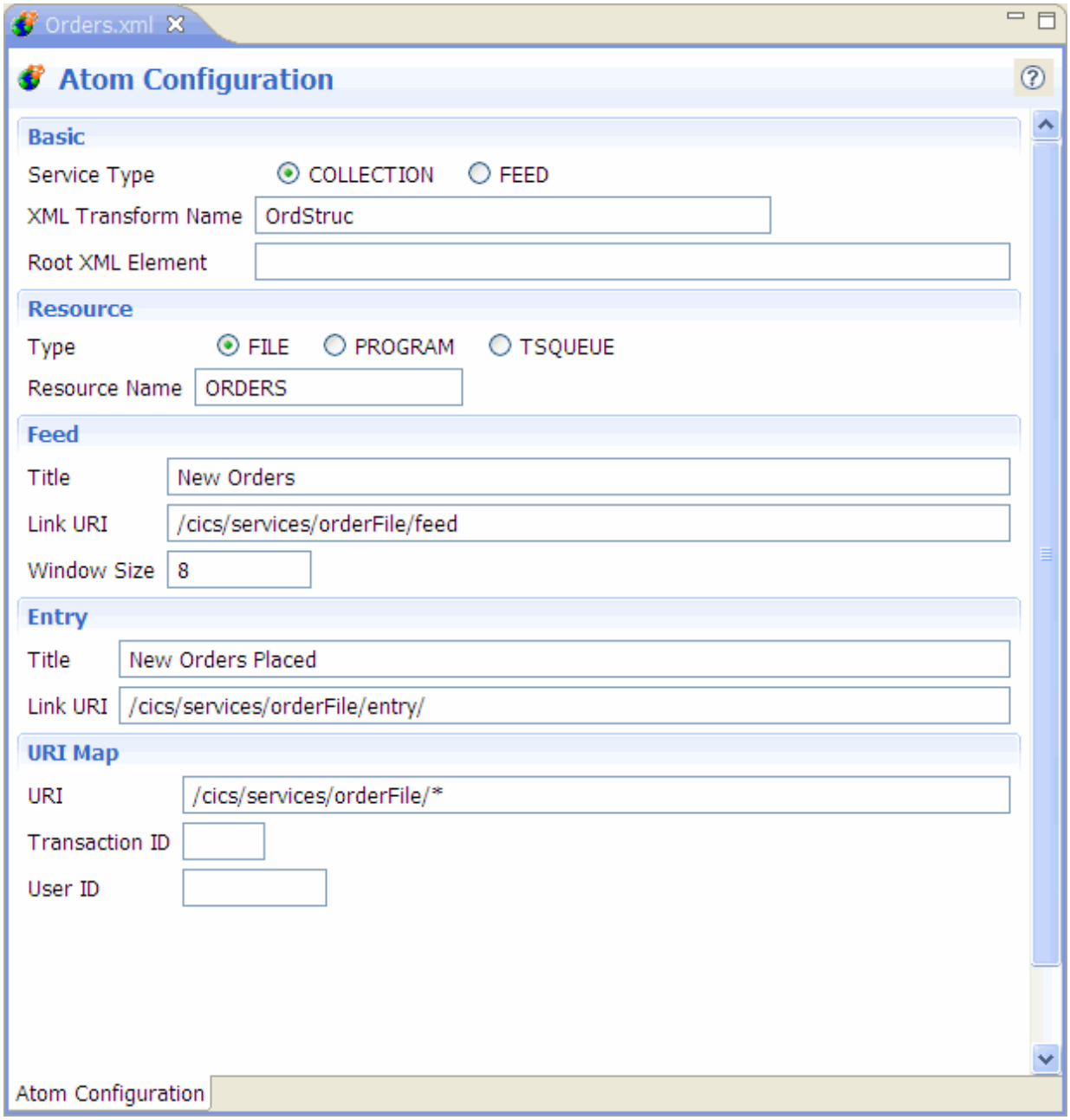
```
       01 ORDER-DETAILS.
        10 ORDER-NUMBER     PIC X(5).
        10 ITEM-ID          PIC X(5).
        10 ITEM-DESC        PIC X(50).
        10 ITEM-QUANT       PIC S9(8) BINARY.
        10 ITEM-PRICE       PIC X(6).
        10 FULFILLED        PIC X.
          88 COMPLETED      VALUE 'Y'.
          88 INCOMPLETE     VALUE 'N'.
      *71 BYTES IN LENGTH
      *KEY IS 5 BYTES FROM OFFSET 0
```

The copy book in this scenario is called OrdStruc.cbl

Procedure

1. If you do not already have a CICS bundle project create one in CICS Explorer.
   a) Switch to the Resource perspective by clicking **Window** > **Open Perspective** > **Other** on the main menu bar. Choose **Resource** from the **Open Perspective** window, and click **OK**.
   b) On the main menu bar, click **Explorer** > **New Wizards** > **Other** > **CICS Resources** > **CICS Bundle project**.
      The Bundle Project wizard opens.
   c) In the **Project name** field, type a name for your new project.
   d) Click **Finish**.

   The new CICS bundle project is listed in the Project Explorer view.
2. Create an XMLTRANSFORM resource using the Import XML Transform Source wizard in CICS Explorer. The XMLTRAMSFORM is created using your copy book.
   For more information, see Creating an XML binding for the Atom feed using CICS Explorer.
3. Create three basic Atom configuration files for the new orders, fulfilled orders, and unfulfilled orders VSAM files in the bundle project, using the Atom configuration wizard in CICS Explorer.
   For more information, see Creating an Atom configuration file. When you have created the Atom configuration file it is displayed in the Atom configuration editor. For this scenario no further editing of the configure files is required. If, however, you want to customize the Atom configuration file by adding additional XML elements or amending the existing data, see Editing an Atom configuration file.
   The Atom configuration file for the new order Atom collection must contain the same data displayed in Figure 6 on page 30. The Atom configuration file for the fulfilled orders Atom collection must be the same as Figure 6 on page 30, however, **Resource Name** is *FULFILL* and **Feed Title** is *Fulfilled Orders*. The Atom configuration file for the unfulfilled orders Atom collection must be the same as Figure 6 on page 30, however **Resource Name** is *UNFULFIL* and **Feed Title** is *Unfulfilled Orders*.
   Figure 6. The Atom configuration file editor in CICS Explorer showing the configuration for the orders file.



Results

An XML binding, schema, and three Atom configuration files are included into a bundle project.

Defining the event processing specification

You use the event binding editor to create an event binding that specifies that CICS must produce an event when records are written to either the fulfilled or unfulfilled VSAM files.

About this task

When a PUT request is made to either the fulfilled or unfulfilled Atom collections, records are added to the fulfilled or unfulfilled VSAM files. You can create an event binding file in the CICS bundle project created in “Creating and installing the Atom collections resources” on page 30 that specifies what events you want CICS to produce when records are written to these VSAM files. The following steps describe how to create an event binding.

Creating an event binding file

You create an event binding file in a CICS bundle project to contain event specifications, their associated capture specifications, the EP adapter used for all events, and options to control the dispatch of events.

About this task

An event can be described in simple words, which should be written by the business manager or analyst who is interested in the event.

Business events are described initially using everyday language. You use the General Information panel in the Event binding editor to describe the business event. For example:

- "I want to know whether our marketing campaign is having the expected impact. Over the next month, give me data twice a week on inquiries and sales of our two new models."
- "There is a wave of credit card fraud from a region. Increase the level of checks for the whole range of transactions from the region. Starting Monday, send a record to a dashboard on my workstation of transactions over 1000 dollars from the region."

An event binding contains control data for one or more events. A business manager uses the Event binding editor to specify the event at a high level.

You specify the events for this binding in the **Event Specifications** tab of the **Event Binding** panel. As a business analyst, you set some values for the event data that you want captured. You repeat this step for each item of data you need during event processing.

Procedure

1. Optional: If you do not already have a project for the event binding, create a new CICS Bundle project in the CICS Explorer.
   a) If you are not already in the Resource perspective, switch to the Resource perspective.
      On the main menu bar, click **Window** > **Open Perspective** > **Other**. Choose **Resource** from the **Open Perspective** window, and click **OK**. The CICS Explorer switches to the Resource perspective.
   b) On the main menu bar, click **File** > **New Wizards** > **Other** > **CICS Resources** > **CICS Bundle project**.
      The Bundle Project wizard opens.
   c) In the **Project name** field, type a name for your new project.
   d) Click **Finish**.

   The new CICS Bundle project is listed in the Project Explorer view.
2. Create a new event binding file in the project for the event binding.
   a) Right-click the project name in the **Project Explorer** view.
   b) From the menu, select **New** > **Event Binding**.
   c) Specify the name of the event binding file.
      The name must not contain embedded spaces.
   d) Click **Finish**.

   A new event binding file is created, and opens in the Event binding editor.
3. Complete the general information describing this event binding
   a) Click the **Event Binding** tab in the Event binding editor.
      The event binding pane is where you specify the general information that describes this event binding.
   b) Describe the event binding.
      The description further identifies this event binding. You might want to describe the set of events that this binding contains.
   c) Specify the **User Tag**.

The user tag further identifies this event binding. For example, you might want to use the user tag to assign a version number to your event bindings. The user tag has a maximum length of 8 characters. The acceptable characters are A-Z, a-z, 0-9, and _. Leading and embedded blank characters are not permitted. The string must not start with 0-9, _, or the string "xml", regardless of whether it is lowercase, uppercase, or mixed case; for example, "Xml" or "xMl".

Results
You can now create specifications and dispatcher information in your event binding file.

Creating an event specification
You add an event specification to an event binding to describe a business event and the information to be captured for the event.

About this task
When you have a clear idea of the events that you want to intercept, you use the Event binding editor to specify them to CICS.

Procedure

1. Click the **Event Binding** tab in the Event binding editor.
2. Click **Add**.

   The **Add Event Specification** window opens.
3. Enter a name for your event specification.

   The acceptable characters are A-Z, a-z, 0-9, and _. Leading and embedded blank characters are not permitted. The string must not start with 0-9, _, or the string "xml", regardless of whether it is lowercase, uppercase, or mixed case; for example, "Xml" or "xMl".
4. Optional: Enter a description for your event specification.
5. Click **OK**.

   A new event specification is listed in the **Event Specifications** section.
6. Double-click the new event specification in the **Event Specifications** section, or click **Edit Details**.

   The **Specification** tab opens with the details of your new event specification.
7. Specify any data that you want to capture in this event. You do not have to capture any data; you might want to emit an event with a name and no data.

   If you do want to capture data, look at the **Emitted Business Information** section in the **Specifications** tab. It contains a table of information that you want to capture for the event. For each item of business information that you want to capture, complete the following steps:

   a) Click **Add**.

      The **Event Information** window appears.

   b) Enter a name for the business information item.

      The acceptable characters are A-Z, a-z, 0-9, and _. Leading and embedded blank characters are not permitted. The string must not start with 0-9, _, or the string "xml", regardless of whether it is lowercase, uppercase, or mixed case; for example, "Xml" or "xMl".

   c) Provide a type, length, and description for the business information item. You can also specify a precision for numeric and scientific business information items.

   After you have added emitted business information items, you can change the order in which the items are emitted by selecting an item and using the **Move Up** and **Move Down** buttons.

Results
You can now create capture specifications to describe your event specification to CICS. You do this by clicking **Add a capture specification**, or, if you are adding a SIGNAL EVENT command to your program, you can generate a capture point for it by clicking **Automatic Capture Specification**.

Adding a capture specification

You add capture specifications to an event specification to create capture points in CICS.

About this task

The Event binding editor guides you through creation of one or more capture specifications. These specifications identify the following items:

- Any point in a program that is considered an event
- The conditions under which the event is considered valid
- How data to be captured at a capture point is to be mapped into the event output data structure

For example, you can specify that any READQ TS (temporary storage READ) command is a potential event. You then qualify the conditions under which the event is raised by adding predicates of the types available for the command that you want to intercept. All predicates must be true for the event to be raised.

Context

The predicates in the capture specification are used for filtering on the context of the capture point.

Every capture point has context items for which you can specify predicates. An example is the transaction ID. All of the available context predicates are shown in the **Filtering** tab. Context predicates are optional; however, you typically specify the transaction ID or current program to emit events from a particular application.

The Context area is used to set filtering options for events. Based on other information that you supply in the Event binding editor, not all fields apply; the context predicates available depend on the capture point. The predicates that you might want to specify are: transaction ID, current program, and user ID.

Event Options

The predicates in the capture specification are used for filtering on the values of event options for a CICS command or system event.

Each capture point can have event options for which you can specify predicates. Some commands have no event options. Event options correspond with the options for a particular EXEC CICS command or system event. An example for the SEND MAP command is the MAP name. An example for the Db2® connection status event is the FROM_CONNECTST status. All the available fixed data values for the capture point are shown in the **Filtering** tab. The format of event options is known to CICS.

Application Data

The predicates in the capture specification are used for filtering on application data that is specified in a CICS command.

Note: You cannot add application data when a capture point of DELETEQ TD or DELETEQ TS is defined for a capture specification.

Application capture points can have variable length data values for which you can specify predicates. The application data corresponds with the options for a particular EXEC CICS command. An example for the SEND MAP command is the FROM field. All the available variable data values for the EXEC CICS command for that capture point are shown in the **Filtering** tab.

Application data predicates are processed by CICS in the order specified; that is, the first predicate is processed, followed by the second predicate, and so on.

Application data is used for storage areas such as COMMAREAs or containers that are passed as options on CICS commands. The format of these areas is not known to CICS, but is known by your application program. You probably have a source language description that you can import to describe the format.

Note: System events do not use application data.

The basic event binding information, entered by the business manager during the high-level definition of the event binding, includes an event specification. This event binding information identified data that was known to be required for further processing, but gave no details of its location. You specify information sources to provide these details.

The Event binding editor helps by presenting in the **Information Sources** tab a table of the information it needs to meet the business specification. The table shows business information from the event specification. For each item in the table you must define the source of information for this capture specification.

Procedure

1. In the **Specification** tab, click an event specification.

   The right pane shows the details of the highlighted event specification.
2. Click **Add a capture specification**.

   The **Add Capture Specification** window is displayed.
3. Enter a name for your new capture specification.

   The acceptable characters are A-Z, a-z, 0-9, and _. Leading and embedded blank characters are not permitted. The string must not start with 0-9, _, or the string "xml", regardless of whether it is lowercase, uppercase, or mixed case; for example, "Xml" or "xMl".
4. Optional: Enter a description for your new capture specification.
5. Click **OK**.

   A new capture specification is created. Three tabs open in the right editor pane: **Capture Point**, **Filtering**, and **Information Sources**. Use each tab to provide information describing the new capture specification.
6. Click the **Capture Point** tab.

   Select the **WRITE FILE** option from the **Capture Point** list.
7. Click the **Filtering** tab.

   Select the **Equals** option from the **Event Options** list and enter `FULFILL` in the **Value** field.
8. Right click the capture specification and select **Copy Capture Specification To...**.
9. Enter a new name in the **Name of copy** field and click **OK**.
10. Open the new capture specification and click the **Filtering** tab.

    Ensure that the **Equals** option in the **Event Options** list is selected and enter `UNFULFIL` in the **Value** field.
11. Click the **Information Sources** tab.

    Perform the following steps on each of the information sources in the table to map the information source to an available data item. The table might contain no information sources if no business data is to be captured on the event.

    You do not have to capture data if it is not required.

    a) Double-click an information source, or select it and click **Edit**.

       The **Event Information Source** window is displayed.

    b) Select an available data item from **Context**, **Data Values**, or **Variable Length Data**.

       If you select a variable length data item, you can either click **Select from imported language structure** and follow the instructions in Selecting variables from imported source code in the CICS Explorer product documentation, or select options for **Location** and **Type** and further define your variable, using the active fields in the panel. Only those fields that are required for the type that you select will be active. For example, if you select the **Character** type, only the **Offset**, **Length**, and **Codepage** fields are active.

       The information source does not have to be the same length as the emitted business information item. CICS pads or truncates the value as necessary.

Results
You have created a capture specification. You can also remove capture specifications by clicking **Remove Capture Specification** in the **Capture Point** tab, and clicking **Yes** to confirm.

Specifying EP adapter and dispatcher information
You specify information in your event binding that controls how CICS emits events produced by the event binding.

About this task
You use dispatcher information to define what happens to events created by this binding. You select the EP adapter to emit events and then select options relevant to the EP adapter.

Procedure

1. Click the **Adapter** tab in the Event binding editor and select the **Use an adapter defined here** option to specify the EP adapter configuration to use for this event binding, the parameters for the EP adapter, and any advanced information.

   You can also choose whether to use a predefined EPADAPTER resource or EPADAPTERSET resource. For more information about a predefined EPADAPTER or EPADAPTERSET resource, see CICS Explorer product documentation.

   If you specify a predefined EPADAPTER or EPADAPTERSET resource, you must enter the name in the **Name** field, or click **Choose** to choose a resource. For more information about choosing adapter resources, see CICS Explorer product documentation.
2. Optional: You can export one or more event specifications in this event binding.
3. Choose the EP adapter type from the **Adapter** list.

   You can specify the following EP adapter types:

   IBM MQ Queue
   Emits events to a IBM MQ message queue in an XML format for consumption by IBM Operational Decision Manager or IBM Business Monitor, or in binary CICS flattened event (CFE) format for consumption directly from IBM MQ.

   For assured event emission, event delivery is assured when the IBM MQ Queue EP adapter is used in combination with persistent IBM MQ message queues.

   Transaction Start
   Emits events to a named CICS transaction. Data is passed to the transaction in a container-based event format. You can specify the CICS system to run the transaction. You can use an existing transaction, if the event data is not required. The transaction start EP adapter does not support assured emission of events.

   TS Queue
   Emits events to a named CICS TS queue in one of the XML formats: Common Base Event, Common Base Event REST, Decision Server Insights Event, or WebSphere Business Events; or in a non-XML (CFE) format. Use this EP adapter to validate that the correct events are being captured with the correct data, and to emit events to any consumer that reads from a TS queue.

   For assured event emission, synchronous transactional events need a recoverable TS queue; synchronous nontransactional events need an unrecoverable queue. Whether TS queues are recoverable depends on the settings of a matching TSMODEL; TS queues are recoverable only when there is a matching TSMODEL. When you use the TS queue adapter for synchronous transactional events, you cannot issue a **DELETEQ TS** command for the event TS queue in the unit of work that captures the event.

   TD Queue
   Emits events to a named CICS transient data queue in one of the following formats: common base event, common base event REST, CICS flattened event (CFE), Decision Server Insights Event, and WebSphere Business Event. Use this EP adapter when developing and testing event specifications to validate that the correct events are being captured with the correct data, and to emit events to any consumer that reads from a TD queue.

For synchronous event emission, the TD queue must be a recoverable intra-partition queue for transactional events and either an unrecoverable intra-partition queue or an extra-partition queue for non-transactional events. The TDQ EP adapter can be used for testing and debugging as well as production.

Custom (User Written)
Emits events in any format that you require. A custom EP adapter is a CICS program that you write to provide a combination of formatting and routing of an event that is not supported by the supplied EP adapters. The custom EP adapter must not carry out any other processing, such as consumption of the event.

For assured event emission, the custom EP adapter must conform to the recoverability requirements of the event. Consult the documentation for your custom EP adapter to see whether it supports transactional or nontransactional synchronous events.

HTTP
Emits events to an HTTP 1.1 compliant server using HTTP POST in XML format for consumption by products such as IBM Operational Decision Manager or IBM Business Monitor. The HTTP EP adapter does not support assured emission of transactional events.

4. Specify the options for your chosen EP adapter type:

- Specify the following options for the WebSphere MQ EP adapter:
  - Specify the queue name of the WebSphere MQ queue on which events emitted by this event binding are placed. You must specify a queue name.
  - Specify whether messages are persistent. Select one of the following values from the **Persistent** list.

    No
    Messages put on the queue by the WebSphere MQ EP adapter are nonpersistent.

    Yes
    Messages put on the queue by the WebSphere MQ EP adapter are persistent.

    Queue Default
    Messages put on the queue inherit the default persistence of the named queue.
  - Specify the message priority. You can either select **Queue Default**, or type a value in the **Priority** field, for the WebSphere MQ message priority, from 0 - 9.
  - Specify the expiry time. You can either select **Never Expire** or type a value for the WebSphere MQ message expiry in the **Expiry Time** field. This time is expressed in tenths of a second. A message becomes eligible to be discarded if it has not been removed from the destination queue before this period elapses.
  - Specify a data format for the event. Select one of the following values from the **Data Format** list:

    CICS Flattened Event (Binary)
    Event data is in a non-XML format.

    Common Base Event (XML)
    Messages are put on the queue in the common base event format required by IBM Business Monitor.

    WebSphere Business Events (XML)
    Messages are put on the queue in the XML format required by the Decision Server Events component of IBM Operational Decision Manager.

    Decision Server Insights Event (XML)
    Messages are put on the queue in the XML format required by Decision Server Insights component of IBM Operational Decision Manager. Event data is in XML format.
- Specify the following options for the Transaction Start EP adapter:
  - Specify the transaction ID of the CICS application that runs as a result of the events. You must specify a transaction ID.

  - Specify a transaction user ID. The transaction that is started by the transaction start EP adapter runs using this user ID.
  - Specify the system ID, which is available only for the Transaction EP adapter. The EP adapter transaction runs on the CICS system with this system ID.
- Specify the following options for the TS Queue EP adapter:
  - Specify the CICS queue name. You must specify a queue name.
  - If your target queue is remote, specify the system ID.
  - Select **Use Auxiliary Temporary Storage** if required.
  - Specify a data format for the event. Select one of the following values from the **Data Format** list:

    CICS Flattened Event (Binary)
    Event data is in a non-XML format.

    Common Base Event (XML)
    Event data is in XML format.

    Common Base Event REST (XML)
    Event data is in XML format.

    WebSphere Business Events (XML)
    Event data is in XML format.

    Decision Server Insights Event (XML)
    Event data is in XML format.
- Specify the following options for the TD Queue EP adapter:
  - Specify the CICS queue name. You must specify a queue name.
  - If your target queue is remote, specify the system ID.
  - Specify a data format for the event. Select one of the following values from the **Data Format** list:

    CICS Flattened Event (Binary)
    Event data is in a non-XML format.

    Common Base Event (XML)
    Event data is in XML format.

    Common Base Event REST (XML)
    Event data is in XML format.

    WebSphere Business Events (XML)
    Event data is in XML format.

    Decision Server Insights Event (XML)
    Event data is in XML format.
- Specify the following options for a Custom (User Written) EP adapter:
  - Specify one of the following:
    - If emission mode is asynchronous the transaction ID for your user-written CICS application that formats, routes, and emits the event. You must specify a transaction ID for asynchronous emission by the custom EP adapter.
    - If emission mode is synchronous the name of your user-written CICS application program that formats, routes, and emits the event. You must specify a program ID for synchronous emission by the custom EP adapter.
  - Write the data to be passed to the Custom EP adapter. Your Custom EP adapter processes this data.
- Specify the following options for the HTTP EP adapter:
  - Specify a URIMAP. The name of a URIMAP resource that represents the connection to the HTTP 1.1 compliant server.
  - Specify a data format for the event. Select one of the following values from the **Data Format** list:

Common Base Event (XML)
Events are sent to an HTTP 1.1 compliant server in the common base event format which is consumable by any product supporting the Common Event Infrastructure.

Common Base Event REST (XML)
Events are sent to an HTTP 1.1 compliant server in the XML format required by IBM Business Monitor.

WebSphere Business Events (XML)
Events are sent to an HTTP 1.1 compliant server in the XML format required by the Decision Server Events component of IBM Operational Decision Manager.

Decision Server Insights Event (XML)
Events are sent to an HTTP 1.1 compliant server in the XML format required by the Decision Server Insights component of IBM Operational Decision Manager.

5. Optional: Specify any required advanced dispatcher options.

   These options are for advanced users, and they control how the EP adapter is run in a CICS system.

   Note: It is generally more efficient to let CICS run the EP adapter under the dispatcher thread. However, you might need the EP adapter to be run as a separate transaction; for example, if you need to run it under a particular user ID that has authority to write to the WebSphere MQ queue, or you want to control the number of concurrent EP adapter tasks by using the TRANCLASS settings.

   a) Specify the emission mode.

      You can specify **ASYNC** or **SYNC** to specify how events are emitted. Specify synchronous emission mode for assured event emission. When the emission mode is synchronous, the WebSphere MQ EP adapter emits these events:

      - Transactional events by using the MQPMO_SYNCPOINT option. WebSphere MQ messages can be recovered in the unit of work for the capturing transaction.
      - Non-transactional events by using the MQPMO_NO_SYNCPOINT option.

      When the emission mode is asynchronous, the WebSphere MQ EP adapter emits events by using the MQPMO_NO_SYNCPOINT option.

      Notes:

      - For synchronous events (assured event emission), the event-capturing application needs write authority to the event emission transport. For asynchronous events, the EP adapter needs write authority.
      - Specifying SYNC NONTRANS for the HTTP EP adapter turns the capturing transaction into a web application that might have to be reconfigured accordingly.
      - Set the DTIMOUT attribute of the TRANSACTION definition for any transaction that could potentially emit synchronous events through the HTTP EP adapter.

   b) Specify the dispatch priority.

      You can specify `Normal` or `High` priority to control how the event dispatcher processes events associated with this event binding. High priority events are emitted as soon as they are available based on the **Events are Transactional** setting. Normal priority events are emitted as soon as they are available based on the **Events are Transactional** setting but after any outstanding high priority events.

      Note: This option does not apply to synchronous emission mode.

   c) Specify the transaction ID and user ID.

      The adapter is initiated with the transaction ID and user ID specified. The transaction ID is not relevant for the Custom EP adapter type as it is set in the Adapter section instead. Depending on the settings on transaction ID and user ID, EP dispatcher task either starts or links to EP adapter program.

Table 3. Transaction ID and user ID for WebSphere MQ EP adapter, TD Queue EP adapter, TS Queue EP adapter, and Transaction Start EP adapter

Transaction ID is specified	User ID is specified	Use Context user ID	EP adapter is	EP adapter runs using	EP adapter transaction ID is
x	x		Attached	Specified user ID	Specified transaction ID
x		x	Attached	Context user ID	Specified transaction ID
	x		Attached	Specified user ID	CEPQ for WebSphere MQ EP adapter CEPR for TD Queue EP adapter CEPT for TS Queue EP adapter CEPS for Transaction Start EP adapter
		x	Attached	Context user ID	CEPQ for WebSphere MQ EP adapter CEPR for TD Queue EP adapter CEPT for TS Queue EP adapter CEPS for Transaction Start EP adapter
x			Attached	CICS default user ID	Specified transaction ID
			Linked	CICS region user ID	Not applicable

If the WebSphere MQ queue, TD queue, TS queue, or the transaction that you want to start is remote, see Security for intercommunication for information about the security settings.

Table 4. Transaction ID and user ID for HTTP EP adapter

Transaction ID is specified	User ID is specified	Use Context user ID	EP adapter is	EP adapter runs using	EP Adapter transaction ID is
x	x		Attached	Specified user ID	Specified transaction ID
x		x	Attached	Context user ID	Specified transaction ID
	x		Attached	Specified user ID	CEPH
		x	Attached	Context user ID	CEPH
x			Attached	CICS default user ID	Specified transaction ID
			Attached	CICS default user ID	CEPH

For the Custom (User-Written) EP adapter which has a transaction ID and an asynchronous emission mode, EP dispatcher task always attaches to this adapter, with the specified user ID, or context user ID , or CICS default user ID.

Note: These two options do not apply to synchronous emission mode.

d) Specify whether events are transactional.

Select the **Events are Transactional** check box if you want CICS to capture events only if the business unit of work (UOW) associated with the event completes successfully.

Note: Consider carefully whether events that are captured from unrecoverable actions should be specified as transactional. For example, if you capture a transactional event from a **WEB SEND** command and the unit of work is then backed out, the event is backed out but the **WEB SEND** command might have caused an HTTP message to be sent.

Clear the **Events are Transactional** check box if you want CICS to process events associated with this event binding outside a transaction. Events are emitted as they are produced.

Results
The adapter and dispatcher information for your event binding is now complete.

Creating an EP adapter file
You create an EP adapter file in a CICS bundle project to contain your EP adapter specification. When an EP adapter is installed as its own resource, you can easily switch events to use a different EP adapter configuration, or different EP adapter altogether, and also send the same event to multiple destinations.

About this task

An EP adapter specification is an XML definition that defines one EP adapter to CICS. An EP adapter consists of adapter and dispatcher information. Use an EP adapter to configure event emission.

There are some situations where you might want the EP adapter to be installed as its own resource. For example:

- Some events have been created in a test environment and sent using an IBM MQ queue to IBM Operational Decision Manager. You now want to deploy these events into production, but in the production environment they must be emitted using a different IBM MQ queue. You can use a separate EP adapter configuration to change the queue name without any changes to the event specifications and associated capture specifications.
- Some events have been emitted using the HTTP EP adapter, but now the volume of events and the requirements for robustness are such that it would be better to use IBM MQ. You can use a separate EP adapter configuration to change the EP adapter without changing the event specifications.

Note: When you use a separate EP adapter configuration you can easily share it between event bindings, resulting in only one resource to manage. When you specify an EP adapter configuration embedded in an event binding, then CICS creates an EPADAPTER resource for you with the same name as the event binding.

You configure this EP adapter in the **Adapter** tab of the **EP adapter** panel. As a system programmer, you set some configuration values for the event data that you want to emit. You repeat this step for each item of data you need during event processing.

Procedure

1. Optional: If you do not already have a project for the EP adapter, create a CICS Bundle project in the CICS Explorer.
   a) If you are not already in the Resource perspective, switch to the Resource perspective.
      On the main menu bar, click **Window** > **Open Perspective** > **Other**. Select **Resource** from the **Open Perspective** window, and click **OK**. CICS Explorer switches to the Resource perspective.
   b) On the main menu bar, click **File** > **New Wizards** > **Other** > **CICS Resources** > **CICS Bundle project**.
      The Bundle Project wizard opens.
   c) In the **Project name** field, type a name for your new project.
   d) Click **Finish**.

   The new CICS Bundle project is listed in the Project Explorer view.
2. Create an EP adapter file in the project you created
   a) Right-click the project name in the **Project Explorer** view.

b) From the menu, select **New** > **EP adapter**.

c) Specify the name of the EP adapter file.

The name must not contain embedded spaces.

d) Click **Finish**.

A new EP adapter file is created, and opens in the Event processing (EP) adapter configuration editor.

3. Complete the general information describing this EP adapter.

a) Click the **Adapter** tab in the Event processing (EP) adapter configuration editor.

The adapter pane is where you specify the general information that describes this EP adapter.

b) Describe the EP adapter.

The description further identifies this EP adapter. You might want to describe the settings that this EP adapter contains.

Results

You can now create specifications and dispatcher information for your EP adapter file.

Creating the order processing application

The order processing application is a Java application. The source code is described.

About this task

The order processing application consists of two Java source files, Main.java and OrderProcessor.java. Main.java makes the GET request to the new orders file and then calls OrderProcessor.java. OrderProcessor.java processes the orders by checking the stock levels of the product ordered, makes the POST requests to either the fulfilled or unfulfilled files. The last section of the OrderProcessor.java makes the DELETE request to the new orders file to delete the original order.

You must create the two source files and copy the Java code into them, compile the files into a .class file and then run the program.

To simplify the order processing application, the stock levels for all products has been set to 100. Code for a dynamic stock levels could be coded, however, it is outside the scope of this scenario.

Procedure

1. Create the Main.java source file and copy the following Java code into it.

```
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class Main {

    /**
     * @param args
     */
    private static String ordersURL = "http://example.com:50001/cics/services/orderFile/
feed";

    public static void main(String[] args) {

        // Create the order processor object
        OrderProcessor worker = new OrderProcessor();
```

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db;
NodeList nodesInOrder = null;
try {
    db = dbf.newDocumentBuilder();
    // Make an XML doc for the ATOM XML for the Orders feed
    Document doc = db.parse(new URL(ordersURL).openStream());

    // Get a list of all entries - each one representing a new order
    NodeList list = doc.getElementsByTagName("entry");
    System.out.println("We have " + list.getLength() + " new orders\n");

    // List of fields we want from the XML order entry
    String order_number = "";
    String item_id = "";
    String description = "";
    String quantity = "";
    String price = "";
    String fulfilled = "";
    NodeList detailList = null;
    for (int i = 0; i < list.getLength(); i++) {
        nodesInOrder = list.item(i).getChildNodes();



        for (int p = 0; p < nodesInOrder.getLength(); p++) {
            Node orderFields = nodesInOrder.item(p);
            // Get the content XML node
            if (orderFields.getNodeName().equals("content")) {
                // Get the OrdStruc XML node
                Node OrdStruc = orderFields.getFirstChild();
                // Get the order_details XML node
                Node details = OrdStruc.getFirstChild();
                // Get all nodes in the order_details node
                detailList = details.getChildNodes();
                for (int n = 0; n < detailList.getLength(); n++) {
                    Node detailItem = detailList.item(n);
                    String nodeName = detailItem.getNodeName();
                    // if we have the order_number
                    if (nodeName.equals("order_number")) {
                        order_number = detailItem.getTextContent();
                        continue;
                    }

                    // if we have the item_id
                    if (nodeName.equals("item_id")) {
                        item_id = detailItem.getTextContent();
                        continue;
                    }
                    // if we have the item_desc
                    if (nodeName.equals("item_desc")) {
                        description = detailItem.getTextContent();
                        continue;
                    }
                    // if we have the item_quant
                    if (nodeName.equals("item_quant")) {
                        quantity = detailItem.getTextContent();
                        continue;
                    }
                    // if we have the item_price
                    if (nodeName.equals("item_price")) {
                        price = detailItem.getTextContent();
                        continue;
                    }
                    // if we have the fulfilled
                    if (nodeName.equals("fulfilled")) {
                        fulfilled = detailItem.getTextContent();
                        continue;
                    }
                    // There is some unrecognised XML nodes...
```

```
                                System.out
                                        .println("Unrecognised segment of XML:");
                                System.out.println(nodeName
                                        + " and the content is :"
                                        + detailItem.getTextContent());

                            }

                        }
                    }

                }

                System.out.println("Order Number:" + order_number);
                System.out.println("Item ID:" + item_id);
                System.out.println("Description:" + description);
                System.out.println("Quantity:" + quantity);
                System.out.println("Item Price:" + price);
                System.out.println("Fulfilled:" + fulfilled);
                System.out.println("\n\n");


                worker.processOrder(order_number, item_id, description,
Integer.valueOf(quantity),
        price,Boolean.getBoolean(fulfilled),nodesInOrder);

        }
        catch(Exception e){

        }

    }

}
```

2. Create the OrderProcessor.java source file and copy the following Java code into it.

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;

import org.apache.http.Header;
import org.apache.http.HeaderElement;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.cache.HeaderConstants;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.InputStreamEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.w3c.dom.NodeList;

public class OrderProcessor {


    private URL successUrl;
    private URL deleteOrderEntry;
    private URL failUrl;


    public OrderProcessor() {

    }

    private void writeToSuccessFile(String order_number, String item_id, String description, int quantity,
 String item_price, boolean fulfilled, NodeList detailList){
        try{
           URLConnection urlConnection;
            DataOutputStream outStream;
            DataInputStream inStream;

           //  Build request body
         String body ="<?xml version=\"1.0\" ?><entry xmlns=\"http://www.w3.org/2005/Atom\">;
         body += "<content><OrdStruc xmlns=\"http://www.OrdStruc.com\"><order_details>:;
         body += "<order_number>"+order_number+"</order_number><item_id>"+item_id+"</item_id>";
         body += "<item_desc>"+description+"</item_desc><item_quant>"+quantity+"</item_quant>";
         body += "<item_price>"+item_price+"</item_price><fulfilled>Y</fulfilled></order_details></OrdStruc></content></
entry>";
```

```
         String bodyLength = new String();
            bodyLength += body.length();
            // Create connection
            successUrl= new URL("http://example.com:50001/cics/services/goodFile/feed");
            urlConnection = successUrl.openConnection();

         ((HttpURLConnection)urlConnection).setRequestMethod("POST");
            urlConnection.setDoInput(true);
            urlConnection.setDoOutput(true);
            urlConnection.setUseCaches(false);
            urlConnection.setRequestProperty("Content-Type", "application/atom+xml;type=entry");
            urlConnection.setRequestProperty("Host", "example.com" );

            // Create I/O streams
            outStream = new DataOutputStream(urlConnection.getOutputStream());
            // Send request
            outStream.writeBytes(body);
            outStream.flush();
            outStream.close();

            inStream = new DataInputStream(urlConnection.getInputStream());

            // Get Response
            // - For debugging purposes only!
            String buffer;
            while((buffer = inStream.readLine()) != null) {
                System.out.println(buffer);
            }

            // Close I/O streams
            inStream.close();
            outStream.close();
        }
        catch(Exception ex) {
            System.out.println("Exception cauught:\n"+ ex.toString());
        }
    }



    public void processOrder(String order_number, String item_id,
                      String description, int quantity, String item_price,
                      boolean fulfilled, NodeList detailList) {

              System.out.println("\n\n==================================");
              System.out.println("Starting to process the order");
              System.out.println("======================================\n\n");

              //we have enough stock
              if(quantity < 100){
                       //add to success file
                       //with fulfilled being true
                       //remove from orders file
                       writeToSuccessFile(order_number,item_id,description,quantity,item_price,fulfilled,detailList);
                       deleteFromOrdersFile(order_number);

              }
              //we do not have enough stock - reject the order
              else{
                       //add to fail file
                       writeToFailFile(order_number,item_id,description,quantity,item_price,fulfilled,detailList);
        }


              System.out.println("\n\n==================================");
              System.out.println("Processed Order");
              System.out.println("======================================\n\n");


    }

    private void writeToFailFile(String order_number, String item_id,
                      String description, int quantity, String item_price,
                      boolean fulfilled, NodeList detailList) {
        // TODO Auto-generated method stub
        try{

        failUrl= new URL("http://example.com:50001/cics/services/failFile/feed");
               URLConnection urlConnection;
             DataOutputStream outStream;
             DataInputStream inStream;

             // Build request body
          String body ="<?xml version=\"1.0\" ?><entry xmlns=\"http://www.w3.org/2005/Atom\">;
          body += "<content><OrdStruc xmlns=\"http://www.OrdStruc.com\"><order_details>:;
          body += "<order_number>"+order_number+"</order_number><item_id>"+item_id+"</item_id>";
          body += "<item_desc>"+description+"</item_desc><item_quant>"+quantity+"</item_quant>";
          body += "<item_price>"+item_price+"</item_price><fulfilled>N</fulfilled></order_details></OrdStruc></content></
entry>";

          String bodyLength = new String();
               bodyLength += body.length();
```

```
            // Create connection

            urlConnection = successUrl.openConnection();

            ((HttpURLConnection)urlConnection).setRequestMethod("POST");
            urlConnection.setDoInput(true);
            urlConnection.setDoOutput(true);
            urlConnection.setUseCaches(false);
            urlConnection.setRequestProperty("Content-Type", "application/atom+xml;type=entry");
            urlConnection.setRequestProperty("Host", "example.com" );

            // Create I/O streams
            outStream = new DataOutputStream(urlConnection.getOutputStream());


            // Send request
            outStream.writeBytes(body);
            outStream.flush();
            outStream.close();

            inStream = new DataInputStream(urlConnection.getInputStream());

            // Get Response
            // - For debugging purposes only!
            String buffer;
            while((buffer = inStream.readLine()) != null) {
                System.out.println(buffer);
              }

            // Close I/O streams
            inStream.close();
            outStream.close();
          }
          catch(Exception ex) {
            System.out.println("Exception cauught:\n"+ ex.toString());
          }
    }

    private void deleteFromOrdersFile(String order_number) {
        // TODO Auto-generated method stub
           try{
            URLConnection urlConnection;
            DataOutputStream outStream;
            DataInputStream inStream;


            // Create connection
            deleteOrderEntry = new URL("http://example.com:50001/cics/services/orders/entry/"+order_number);
            urlConnection = deleteOrderEntry.openConnection();

            ((HttpURLConnection)urlConnection).setRequestMethod("DELETE");
            urlConnection.setDoInput(true);
            urlConnection.setDoOutput(true);
            urlConnection.setUseCaches(false);
            urlConnection.setRequestProperty("Host", "example.com" );


            inStream = new DataInputStream(urlConnection.getInputStream());

            String buffer;
            while((buffer = inStream.readLine()) != null) {
                System.out.println(buffer);
            }

            // Close I/O streams
            inStream.close();
        }
        catch(Exception ex) {
            System.out.println("Exception cought:\n"+ ex.toString());
        }

    }
}
```

3. Compile the source files into a .class file
4. Run the order processor application

Results

An order processing application is created that can obtain order information from the new orders Atom collection, process the order, make a POST request to either the fulfilled or unfulfilled Atom collection, and make a DELETE request to delete the original order.

Chapter 5. Troubleshooting for event processing

Use this information to help you identify the source of errors that can affect event processing.

Event processing problem determination tools

Use these problem determination tools to diagnose the cause of event processing problems.

Statistics utility transaction STAT

You can review event processing and CAPTURESPECs statistics that are relevant to CICS event processing problem determination using the STAT transaction in sample group DFH£STAT.

Statistics utility output

The CAPTURESPECs option of the STAT transaction generates the CAPTURESPEC report. For more information, see CAPTURESPEC report.

The EPADAPTER option of the STAT transaction generates the EPADAPTER report. For more information, see EPADAPTER report.

The `Event Binding` option of the STAT transaction generates the EVENTBINDING report. For more information, see EVENTBINDING report.

The `Event Processing` option of the STAT transaction generates the EVENTPROCESS report. For more information, see EVENTPROCESS report.

Level 1 and level 2 trace from CICS event capture 'EC' component

You can determine entry and exit points from the event capture process using level 1 trace. You can determine all the stages of event point filter evaluation and event capture using a level 2 trace.

Event capture level 1 trace

Level 1 trace records entry to and exit from the event capture process at an event point as follows:

```
AP 3530 ECEC ENTRY - FUNCTION(EVENT_CAPTURE) PARM_LIST(2AD09BA0) EVP_ADDRESS(294C214A) :CMD-POST(SIGNAL_EVENT)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:29.6997843842 INTERVAL-00.0000007187
=000585=
           1-0000  00300000 0000026B 00000000 00000000  AB000000 00000000 01000100 2AD09BA0
*......,.....................}..*
             0020  00000000 00000000 294C214A 00000000
*.........<.$....                *
           2-0000  28020000 02280200 02010020 00000000  2B9A3420 000CE2C9 C7D5C1D3 6DC5E5C5
*.....................SIGNAL_EVE*
             0020  D5E30000 00000000 00000000 00000000  05C5E5C5 D5E30000 00000000 00
*NT..............EVENT.......   *
…
AP 3531 ECEC EXIT  - FUNCTION(EVENT_CAPTURE) RESPONSE(OK) EVENTS(1)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:29.7011492756 INTERVAL-00.0000005937
=000655=
           1-0000  00300000 0000026B 00000000 00000000  AB000000 00000000 01000100 2AD09BA0
*......,.....................}..*
             0020  00000000 00000000 294C214A 00000001
*.........<.$....                *
```

The parts of most use in problem determination are the event point on entry `CMD-POST(SIGNAL_EVENT)` and the number of events captured on exit `EVENTS(1)`.

Event capture level 2 trace

Level 2 trace records all the stages of event point filter evaluation and event capture. It is particularly useful to see why a filter evaluated to true or false, and to see the data that was captured.

When the event point has filters on the primary predicate, a match or non match is traced:

```
AP 353C ECEC EVENT - PRIMARY_PREDICATE MATCHED

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-A99F44AA TIME-11:02:29.6961430400 INTERVAL-00.0000003593
=000444=
           1-0000  03000000 E3C5E2E3 F1404040 00000000  00000000 00000000 00000000 00000000
*....TEST1   ....................*
           0020  00000000
*....                          *
```

When evaluation of the filter of a CAPTURESPEC starts:

```
AP 353B ECEC EVENT - FILTERING CAPTURESPEC(Prog_Init_capturespec)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-A99F44AA TIME-11:02:29.6961442431 INTERVAL-00.0000012031
=000445=
           1-0000  00A86EC4 C6C8C5C3 C3E24040 40404040  D7999687 6DC99589 A36D8381 97A3A499  *.y>DFHECCS
Prog_Init_captur*
           0020  85A29785 83404040 40404040 40404040
*espec                         *
```

For each filter predicate, the result (true or false) is traced. Here, an EVENT parameter is tested:

```
AP 3534 ECEC EVENT - PREDICATE_TRUE - TESTING PARAMETER(EVENT) EQUAL

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:29.6997860092 INTERVAL-00.0000003437
=000587=
           1-0000  006D6EC4 C6C8C5C3 C6D74040 40404040  00000000 00010402 80058000 00000000
*._>DFHECFP     ................*
           0020  00007780 00000000 00000005 C5E5C5D5  E3404040 40404040
*...........EVENT              *
           2-0000  C5A58595 A3
*Event                         *
           3-0000  C5A58595 A3
*Event                         *
```

Here, data in a FROM parameter is tested. The result is TRUE because the test was NOT_EQUAL, the FROM parameter (trace item 2) field was blanks, and the tested for value (trace item 3) was SOMEVALUE:

```
AP 3534 ECEC EVENT - PREDICATE_TRUE - TESTING BYTES IN PARAMETER(FROM) NOT_EQUAL

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:29.7020589155 INTERVAL-00.0000004375
=000697=
           1-0000  00916EC4 C6C8C5C3 C6D74040 40404040  00000000 00180802 40050080 0C022005
*.j>DFHECFP      ........ .......*
           0020  00808780 0000001B 00000009 C6D9D6D4  40404040 40404040
*..g.........FROM              *
           2-0000  40404040 40404040 40
*                              *
           3-0000  E2D6D4C5 E5C1D3E4 C5
*SOMEVALUE                     *
```

Here, the result is FALSE because the field offset (trace item 2) and the field length (trace item 3) take it beyond the length of the FROM parameter:

```
AP 3536 ECEC EVENT - PREDICATE_FALSE - NO FIELD FOR BYTES IN PARAMETER(FROM)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:30.9828421982 INTERVAL-00.0000003593
=001713=
           1-0000  00916EC4 C6C8C5C3 C6D74040 40404040  00000000 00180802 40050080 0C022005
*.j>DFHECFP     ........ .......*
           0020  00807780 00000064 00000009 C6D9D6D4  40404040 40404040
*...........FROM               *
           2-0000  00000064
*....                          *
           3-0000  00000009
*....                          *
```

Here, the result is FALSE because zoned decimal data (trace item 2) is invalid:

```
AP 3538 ECEC EVENT - PREDICATE_FALSE - BAD DATA IN ZONED IN PARAMETER(INTO-SET)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:30.9827011962 INTERVAL-00.0000059218
=001689=
           1-0000  00C76EC4 C6C8C5C3 C6D74040 40404040  2B9A4140 001E0802 40050180 00000000
*.G>DFHECFP     ... .... .......*
           0020  00008780 00000005 0000001F C9D5E3D6  60E2C5E3 40404040                  *..g.........INTO-
SET            *
           2-0000  40404000 57404040 40404040 40404040  40404040 40404040 40404040 404040
*   ..                         *
           3-0000  F1F2F3F4 F5F6F7F8 F9F0F1F2 F3F4F5F6  F7F8F9F0 F1F2F3F4 F5F6F7F8 F9F0D1
*12345678901234567890123456789 0J *
```

If all the filter predicates for a CAPTURESPEC are TRUE, event capture starts:

```
AP 353E ECEC EVENT - CAPTURING_EVENT CAPTURESPEC(EXEC_DELETEQ_TS)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:30.9857027761 INTERVAL-00.0000005781
=001741=
           1-0000  00A86EC4 C6C8C5C3 C3E24040 40404040  C5E7C5C3 6DC4C5D3 C5E3C5D8 6DE3E240  *.y>DFHECCS
EXEC_DELETEQ_TS *
             0020  40404040 40404040 40404040 40404040
*                              *
```

The capture data containers are traced so that the data (trace item 2) can be checked:

```
AP 3533 ECEC EVENT - CAPTURE_CONTAINER CONTAINER(DFHEP.DATA.00001)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80083AA4 TIME-11:02:30.9868634948 INTERVAL-00.0000003750
=001771=
           1-0000  C4C6C8C5 D74BC4C1 E3C14BF0 F0F0F0F1
*DFHEP.DATA.00001              *
           2-0000  D4E8D8E4 C5E4C5F1 F2404040 40404040
*MYQUEUE12                     *
```

If capture data is not available, that fact is traced too:

```
AP 353A ECEC EVENT - UNAVAILABLE_DATA IN(CHANNEL)

         TASK-00057 KE_NUM-0076 TCB-QR   /009BAA68 RET-80082918 TIME-11:02:35.2281765703 INTERVAL-00.0000006562
=002288=
           1-0000  C3C8C1D5 D5C5D340 40404040
*CHANNEL                       *
```

For information about using CICS debugging tools, trace, and dump, see Troubleshooting and support.

Formatted dump for EC component

The EVENT BINDING SUMMARY in the EC formatted dump is particularly useful because it lists the details of the CAPTURESPECs.

```
==EC: EVENT BINDING
SUMMARY

 KEY FOR
SUMMARY

   P1/P2 = Parameter offset in parameter
list
   AB    = Argument existence bit byte
position
   AM    = Argument existence bit
mask
   KB    = Keyword existence bit byte
position
   KM    = Keyword existence bit
mask
   MN    = Meaning of keyword existence
bit
=ECEVB: MyFirstEventBinding             UserTag=VO_9     EPAdapter=MyFirstEventBinding
Enabled=Y
  ECCS: EXEC_SEND_MAP                   SEND_MAP(postcmd) group/function=1804
Events=0000000000000003
        Specific Primary_Predicate: MAP='DFHCTMO'
CapturesFailed=00000000
         Filter type     Keyword      Op  Offset   Length   P1 AB AM KB KM MN P2 AB AM KB KM MN Value(up
to 32 chars)
   ECFP: data_value      MAPSET       EQ  00000000 00000007 10 02 10 09 01 00 00 00 00 00 00 00
'DFHCTRM'
   ECFP: keyword         ALARM        EXS 00000000 00000000 00 00 00 06 04 00 00 00 00 00 00
00
   ECFP: easydata_bytes  FROM         NEQ 0000001B 00000009 08 02 40 05 00 80 0C 02 20 05 00 80
'SOMEVALUE'
   ECCD: data_value      MAP              00000000 00000007 04 02 80 05 00 00 00 00 00 00 00
00
   ECCD: data_value      MAPSET           00000000 00000007 10 02 10 09 01 00 00 00 00 00 00
00
   ECCD: fulldata        FROM             0000001B 00000010 08 02 40 05 00 80 0C 02 20 05 00 80
```

The event binding summary shows an EVENTBINDING called MyFirstEventBinding. The EVENTBINDING has one CAPTURESPEC called EXEC_SEND_MAP that captures events following a SEND

MAP command, and has captured three events so far. It has a specific primary predicate, which is that the MAP parameter must equal DFHCTM0. There are three further filter predicates: the MAPSET parameter must equal DFHCTRM, the ALARM parameter must be present, and the 9 bytes of data at offset 27 in the FROM parameter must not be SOMEVALUE. If the filter evaluates to true, an event is captured and will include three items of capture data: the value of the MAP parameter, the value of the MAPSET parameter, and the 16 bytes of data at offset 27 in the FROM parameter.

For information about using CICS debugging tools, trace, and dump, see Troubleshooting and support.

Master terminal CEMT INQUIRE transaction

You can check the status of event processing, list the installed EVENTBINDINGs and their status, and see the temporary storage queues created by the TS Queue EP adapter or the sample custom EP adapter.

You can use the CEMT INQUIRE transaction to perform problem determination as follows:

- CEMT INQUIRE EVENTPROCESS to check the status of event processing.
- CEMT INQUIRE EVENTBINDING to list the installed EVENTBINDINGs and their status.
- CEMT INQUIRE TSQUEUE to see the temporary storage queues created by the TS Queue EP adapter or the sample custom EP adapter.

You can also use the IBM CICS Explorer views to perform problem determination as follows:

- Event processing view to check the status of event processing.
- Event bindings view to list the installed EVENTBINDINGs and their status.
- TS queues view to see the temporary storage queues created by the TS Queue EP adapter or the sample custom EP adapter.

TD Queue or TS Queue EP adapter and sample custom EP adapter

The TD and TS Queue EP adapter write CICS event objects to a TD or TS queue in CICS flattened event, common base event, common base event REST, Decision Server Insights Event, and WebSphere Business Event formats. The sample custom EP adapter writes events to a TS queue in a simplified flattened event format. You can use these EP adapters during development of event processing applications, or during problem determination, to check the number of events and the contents being captured.

Event processing problem determination procedures

Problems can occur when expected events are not captured, unexpected events are captured, capture data is missing or incorrect, captured events are missing, or event data is replaced by asterisks (*).

Expected events not captured

Work out why expected events have not been received by an event consumer.

Here are some possible reasons why expected events are not captured:

- Check that Event Processing (EP) is started with the **CEMT INQUIRE EVENTPROCESS** command. When the status is not ‘started' then set it to ‘started' and try to capture the events again.

  You can also use CICS Explorer or the CICSPlex SM Web User Interface to check that EP is started.
- Using the event binding editor, find the names of the EVENTBINDING and CAPTURESPEC that are intended to capture the events. Also note the capture points used.
- Check that the event binding is active with the **CEMT INQUIRE EVENTBINDING** command. When the EVENTBINDING is not installed, it might be that the installation failed. Check the CICS message log for message DFHEC1003, which is present for each event binding that fails to install. The DFHEC1003 message is typically preceded by another message describing the cause of the failure. In that case, correct the EVENTBINDING, redeploy the bundle containing it and try to capture the events again.

You can also use tCICS Explorer or the CICSPlex SM Web User Interface to check event bindings.

- Obtain the CAPTURESPEC statistics. Check the count of events for the CAPTURESPEC concerned. When the count shows that the required number of events are being captured, see "Captured events missing" on page 52.
- Obtain a level 1 and 2 trace for EC, and search it for the capture points where events are required. The best way to search depends on the situation. One approach is to search the trace for ECEC ENTRY trace records for the required capture point, for example CMD-POST(PUT_CONTAINER) that are followed by ECEC EXIT trace records with EVENTS(0).
- When the required capture point has no ECEC ENTRY trace records, obtain a dump formatted for EC to check that the required CAPTURESPEC entry ECCS: is present. If the ECCS: entry is not present, the capture specification is missing from the event binding. Edit the event binding to add it, redeploy the bundle containing it, and try to capture the events again. When the required CAPTURESPEC entry is present the application program is probably not reaching the expected capture points and you must review the program and redesign your event bindings appropriately. You can check this situation by using level 2 trace for the EI component, the CICS execution diagnostic facility (CEDF), or both.
- When ECEC ENTRY trace records for the required capture point are present, work through the ECEC EVENT trace records to find the record with PREDICATE_FALSE. The record with PREDICATE_FALSE explains why no event was captured at this capture point. All predicates must be true for an event to be captured.
- When system events are missing, check the CICS message log for messages DFHEC1023 and DFHEC1024. Transactional and synchronous event emission is not supported for system events.
- When unhandled transaction abend system events are missing, check that the abend is not getting handled for you by CICS. For example, any commands issued from CECI are caught by the EXEC CICS HANDLE ABEND issued by the CECI transaction at the start of the session. Similarly, abends in CICS initialization and shutdown PLT programs are caught by the CICS code that processes the PLT. For a CICS web support transaction, any abends are caught by the EXEC CICS HANDLE ABEND issued by the CICS web support alias program DHWBA. For more information about CECI, see How CECI runs.

Unexpected events captured

Work out why more events than expected have been received by an event consumer.

- One possibility to bear in mind during the investigation is that the events might have been captured correctly but routed to the wrong adapter, event consumer, or both.
- Obtain the Event Processing and CAPTURESPEC statistics. Check the number of events processed by each EP adapter to see which has received more than expected. When more than one custom adapter is in use, use the **CEMT INQUIRE PROGRAM** command to determine how many events were processed by each adapter and which one received too many events.
- Obtain a dump formatted for EC and EP. Check the =ECEVB: lines in the *Event Binding Summary* to discover which event bindings and capture specifications are sending events to the adapter. For more details about the adapter, for example, adapter type, transaction ID and so on, see the *Event Processing Adapters* summary in the EP section of the dump.
- Obtain a level 1 & 2 trace for EI and EC components. Search the trace for ECEC EVENT - CAPTURING_EVENT CAPTURESPEC(xxxx where xxxx is the required CAPTURESPEC name. Each occurrence in the trace represents an event captured because of that CAPTURESPEC. Examine the preceding trace entries with ECEC EVENT - PREDICATE_TRUE. These entries explain why the event filter was passed and the event captured.

Capture data is missing or incorrect

A capture specification can specify one or more items of data to be captured with an event; however, there are situations that can cause data items to be missed in the capture.

For example, any of the following circumstances could cause data items to be missed:

- The data item is from an optional parameter that is not present.
- The data item is from a channel or container that is not present.

- The data item is from a container, communication area (COMMAREA), or data area that is too short to contain the data item.

In each of these cases, a level 2 trace for EC contains an ECEC EVENT trace record with UNAVAILABLE_DATA and the name of the source of the data. Examine earlier trace records to determine the relevant capture specification (CAPTURESPEC) and capture point. Perhaps an event is not required if that capture data is unavailable, in which case you can add a suitable predicate to the filter and redeploy the bundle that contains the CAPTURESPEC. However, if the events are required and some capture data is not available, you must configure the event consumer to handle this situation correctly.

Incorrect capture data can be caused by these situations:

- Incorrect filtering. If your capture specification is too generic, you might be trying to capture data from commands that you did not intend to enable for events, or from records whose layout does not match the capture specification. For example: you might be filtering on all **WRITE FILE** commands where the file name starts with AB but where you did not intend to capture data from the write to file ABC. Or when capturing events on **WRITE FILE (ABC)** and more than one record format is written to the file, you might have to filter on the record format to extract data items from the record correctly. A level 1 and 2 trace for EI and EC components can help to identify this situation.
- Incorrect capture data specification. For example: the data is being captured from the wrong parameter or container, or with the wrong offset, length, or both. In this situation, a dump formatted for EC to check the required capture data item ECCD: is useful.

Captured events missing

Events have been captured, but not emitted, and they have not been received at the point where they were expected.

- Obtain the event processing statistics and look to see if any events have been lost by the dispatcher or adapters. When the events lost reason is config, examine the CICS message log for explanations. Possible reasons include incorrect adapter transaction ID and incorrect or revoked password. Correct these problems appropriately. When the events lost reason is other, examine the CICS message log for explanations. Possible reasons include CICS short on storage. Take the appropriate action. When no explanatory message is found, check the CICS exception trace entries for an explanation.
- Check the statistics for transactional events that have been discarded because of transactions being backed out. Check if the number matches the number of missing events.
- Check the number of times the adapter has been started using the **CEMT INQUIRE PROGRAM** command.
  - When the adapter has not been started for all the events, the event has been lost by the EP dispatcher.
  - When the adapter has been started the required number of times the event might have been lost by the adapter, by the transport to the event consumer, or by the event consumer.
- When event processing is stopped, or a CICS immediate shutdown is performed, event capture is stopped and in some cases captured events are not emitted.
- DFHECnnnn messages and AECx abend codes can help you diagnose problems with your EP adapters.
- If your adapter transaction is subject to scheduling constraints, then an AKCC abend for the adapter transaction could indicate the cause of your missing event.

Event data replaced by asterisks (*)

The data in emitted XML or in the DFHEP.CHAR containers is replaced by one or more asterisks (*) in these situations.

- The data is numeric and too large for the formatted field width.
- The captured data is missing. For more information, see “Capture data is missing or incorrect” on page 51.

Unexpected transaction abend ASP7

The unit of work that an event originated from is backed out with transaction abend code ASP7 when a synchronous event cannot be emitted.

Transaction abend (abnormal end) code ASP7 can be logged for reasons other than event processing, so the first task is to find out whether event processing caused the event to end abnormally.

- Check for exception trace entry AP 05AE, which is displayed in the log shortly before the abnormal end. If this exception trace entry is not displayed, the abnormal end was caused by something other than event processing.
- If exception trace entry AP 05AE is logged, check the CICS message log for message DFHEC1022; this message contains the name of the related event binding.

  Note: For long-running transactions, message DFHEC1022 might appear long before transaction abend code ASP7 because the message occurs only when the first synchronous event in the unit of work cannot be emitted, while the abend code occurs only when the unit of work is committed.
- If you do *not* expect the named event binding to cause events to be captured from the failing unit of work, review the filters of the CAPTURESPECS in the event binding. Change the filters to prevent the unwanted events and deploy the changed event binding to CICS.
- If you expect the named event binding to cause events to be captured from the failing unit of work, examine the CICS message log and the CICS exception trace just before message DFHEC1022 to see why the event was not emitted.

  Exception trace entry 040A indicates that the link to the EP adapter failed and why. One possibility is that the EP adapter program is not enabled or defined to CICS. Another possibility is an EP adapter abend code, which indicates that the EP adapter detected an error. When the EP adapter detects an error, the IBM-supplied EP adapters generate both messages and exception trace entries to describe the problem in more detail.

  Common problems that are detected by EP adapters:

  – Message queues are not defined.

  – A message queue is defined with characteristics that are incorrect for the type of event.

  Correct the problem that is indicated and run your program again to capture the events correctly.

Diagnosing deployment errors

Deployment errors can occur when you use CICS Explorer or CICSPlex SM to deploy bundles, or install resources in CICS. The most common deployment errors are described, including the symptom of the problem, the cause and the solution.

About this task

If a deployment error occurs, Bundle resources typically install in a DISABLED state. Information and error messages associated with installing resources are located in the system log.

Procedure

- You receive a DFHEC1003 error message from CICS:

```
DFHEC1003 08/23/2010 17:31:58 IYK3ZMC1 The CICS event capture component failed to create the
EVENTBINDING resource
           MyEPCatalogSampleEventBinding in BUNDLE MYTEST because the XML data in the event binding
could not be parsed.
```

  An error has occurred creating event binding *evbname* in bundle *bundle*. This is probably caused by an error or inconsistency in the event binding XML. If the event binding was built by the event binding editor, this might indicate an error in CICS code. To resolve this issue:

  a) Inspect the CICS trace and EC domain message log for any related trace entries or messages.

  b) Validate the event binding against the event processing schema for the CICS system into which the bundle is being installed.

c) Correct the event binding, discard the bundle, and reinstall it.

- You receive a DFHEC1013 error message from CICS:

```
DFHEC1013 08/23/2010 17:17:22 IYK3ZMC1 The CICS event capture component failed to create the
EVENTBINDING resource
           MyEPCatalogSampleEventBinding in BUNDLE MYTEST because the event binding schema level is not
supported: 0900.
```

The event binding schema level is not supported and the EVENTBINDING resource cannot be created.

This is probably caused by an error or inconsistency in the event binding XML. If the event binding was built by the event binding editor, this might indicate an error in CICS code. To resolve this issue:

a) Inspect the CICS trace and EC domain message log for any related trace entries or messages.

b) Validate the event binding against the event processing schema for the CICS system into which the bundle is being installed.

c) Correct the event binding, discard the bundle, and reinstall it.

- You receive a DFHPI1007 error message from CICS:

```
DFHPI1007 08/23/2010 17:31:58 IYK3ZMC1 00165 XML to data transformation failed because of incorrect
input
(XML_FORMAT_ERROR PI not closed) for EVENTBINDING MyEPCatalogSampleEventBinding.
```

CICS has failed to convert some XML data into application data. This XML may be the body of a SOAP message received from a partner process. The reason for the failure is due to a problem with the content of the XML. The XML is either not well formed, invalid with respect to the XML schema or does not conform to one of the internal constraints of the CICS XML transformation service. An *error_qualifier* may be provided to help identify the source of the problem. In some cases the *error_qualifier* will be empty. To resolve this issue:

a) Examine the exception trace entry for further information.

b) Consider using the WEBSERVICE or XMLTRANSFORM validation option to test that the XML is valid for the schema. XML is case sensitive. Ensure that the XML element, attribute, and namespace names used within the XML are correct with respect to the schemas that describe the XML.

c) Correct or change the partner process to ensure that the XML sent to CICS is appropriate to be consumed by CICS.

d) If a SOAP message is changed by a handler program as part of the CICS PIPELINE processing then ensure that the handler has not introduced this problem.

- You receive a DFHRL0102 error message from CICS:

```
DFHRL0102 E 08/23/2010 17:17:22 IYK3ZMC1 CEDA The CICS resource life-cycle manager failed to create the
resource
           MyEPCatalogSampleEventBinding and returned with reason CALL_BACK_ERROR.
```

The CICS resource lifecycle resource class, DFHRLRS, failed to create the resource *resource_name*.

To resolve this issue:

a) Correct the problem identified in the message.

b) If the message reports that no further information is available, refer to trace.

Chapter 6. Event processing adapters and formats

CICS provides a number of event processing (EP) adapters that format and then emit events from a CICS system. Alternatively, you can write a custom EP adapter. You can use a number of event processing formats to format and emit your events.

Event processing adapters

CICS event processing (EP) adapters are programs that format and then emit events from a CICS system. EP adapters format event data into a suitable output format and route the event using a transport mechanism that makes it available to potential event consumers.

You can use CICS EP adapters to emit business events to your consumers by using these transport mechanisms:

- IBM MQ, in XML format for consumption by IBM Operational Decision Manager or IBM Business Monitor, or in binary CICS flattened event (CFE) format for consumption directly from IBM MQ
- HTTP POST, in XML format for consumption by IBM Operational Decision Manager, IBM Business Monitor, or any other event processor that accepts XML events using HTTP.
- A CICS temporary storage queue
- A CICS transaction
- A CICS transient data queue

You can also implement a user-written or vendor-written custom EP adapter for specific user or vendor business requirements.

Any business events that are captured from a CICS application or system are processed and then dispatched by the event dispatcher to the EP adapter specified for that event, for formatting and routing to the relevant processor or consumer.

You can cause the same event to be directed to more than one EP adapter by creating an EP adapter set containing the names of multiple EP adapters. You can specify the EP adapter set name in one or more event bindings.

You specify business events and the CICS EP adapters that they use by using the Event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

Figure 7 on page 56 illustrates the CICS EP adapter data flow from capture point to the final event consumer.

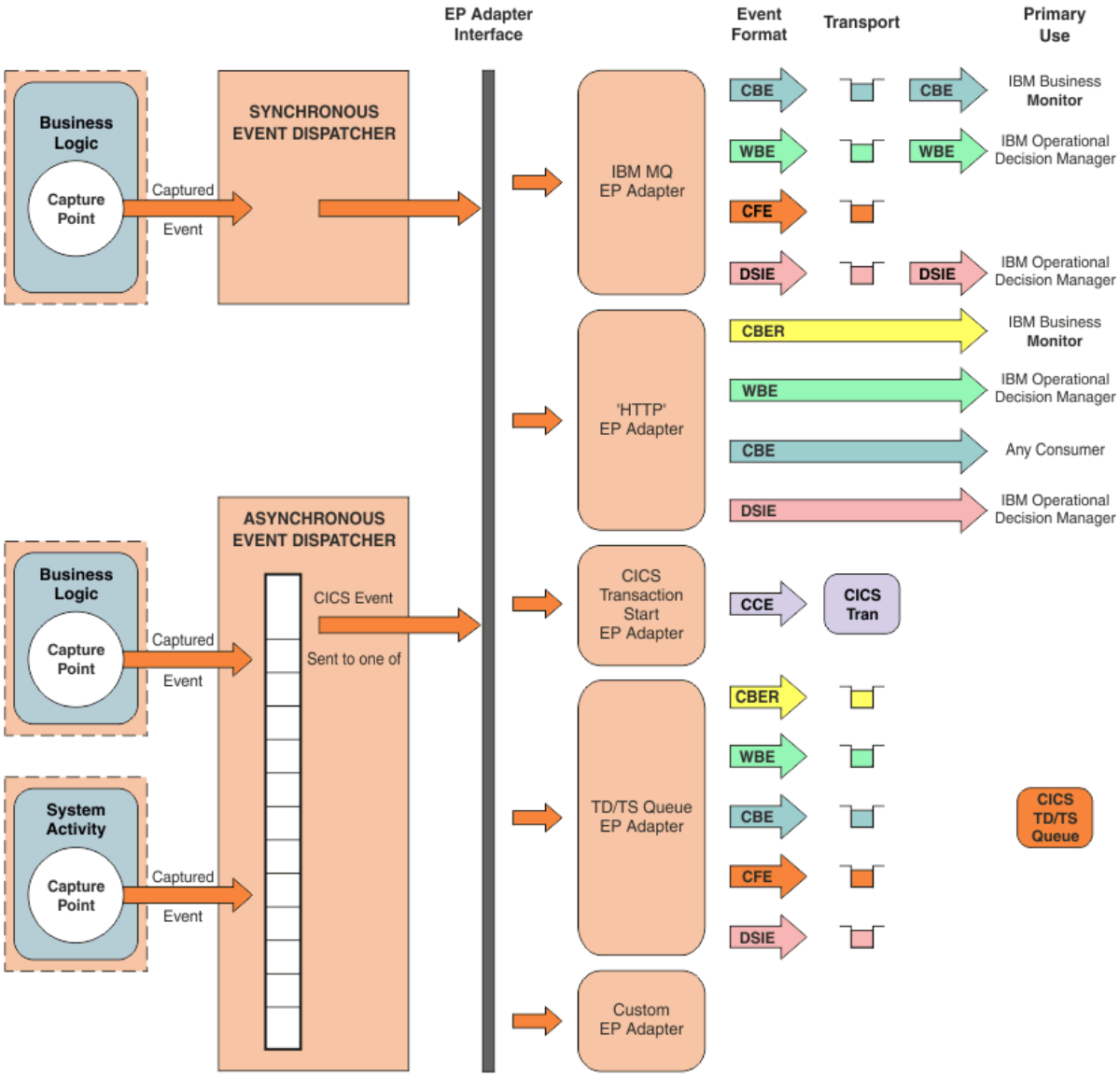


Figure 7. Conceptual view of the CICS EP adapter data flow

You specify the required format and transport in the EP adapter configuration of the event binding, so that all events in a binding are formatted and routed in the same way.

CICS provides some EP adapters. Alternatively, a user or vendor can write a custom EP adapter, which can be modified to suit your requirements.

Because only certain combinations of format and transport are valid, or supported, with the EP adapters that are supplied by CICS, you select the EP adapter for the required transport and then, where appropriate, select the format.

Table 5 on page 57 shows the transport mechanisms and message formats that are available.

Table 5. EP adapter transports, formats, and intended consumers

CICS event binding editor: adapter specification	Event Processing adapter used	Transport for emission	Event format emitted	Intended consumer
Message queue	IBM MQ	IBM MQ	Common base event	IBM Business Monitor and any application that consumes a Common Base Event
			CICS flattened event	An application that gets the CICS flattened event (CFE) from the message queue and consumes it by using the mappings that are provided by these copyboooks: • The DFHEPFE* copybooks in either SDFHMAC or SDFHC370 • The copybook that is exported for the CFE event.
			Decision Server Insights Event	Decision Server Insights component of IBM Operational Decision Manager
			WebSphere Business Event	Decision Server Events component of IBM Operational Decision Manager
HTTP	HTTP	HTTP	Common base event	Any application that consumes a Common Base Event
			Common Base Event REST	IBM Business Monitor or any application that consumes a Common Base Event REST
			Decision Server Insights Event	Decision Server Insights component of IBM Operational Decision Manager
			WebSphere Business Event	Decision Server Events component of IBM Operational Decision Manager
TD queue	TDQ	A transient data record	Common base event	A CICS application that consumes the event from the transient data queue (TDQ). The TDQ EP adapter can be used for testing and debugging as well as production.
			Common Base Event REST	
			CICS flattened event	
			Decision Server Insights Event	
			WebSphere Business Event	
Transaction Start	Transaction start	CICS containers	CICS container-based event	A CICS application that consumes the event data from containers

Table 5. EP adapter transports, formats, and intended consumers (continued)				
CICS event binding editor: adapter specification	**Event Processing adapter used**	**Transport for emission**	**Event format emitted**	**Intended consumer**
TS queue	TSQ	A temporary storage record	Common base event	A CICS application that consumes the event from the temporary storage queue (TSQ). The TSQ EP adapter is generally used for testing and debugging, not production.
			Common Base Event REST	
			CICS flattened event	
			Decision Server Insights Event	
			WebSphere Business Event	
Custom	User-written or vendor-written. With a custom EP adapter, the transport mechanism and message format are determined by the custom EP adapter, allowing any transport mechanism that can be accessed from a CICS program to be used and any required format to be supported.			

Table 6 on page 58 shows the combinations of emission modes and transactional modes that are supported for each EP adapter. The letter "x" indicates a supported combination of modes.

Table 6. EP adapter emission and transactional modes					
Event processing adapter	**Emission mode**		**Transactional mode**		**Notes**
	SYNC	**ASYNC**	**TRANS**	**NONTRANS**	
IBM MQ	x		x		Event emission is assured by using synchronous emission mode. For the IBM MQ EP adapter, event delivery is assured when synchronous emission mode is used in combination with persistent queues.
	x			x	
		x	x		
		x		x	
HTTP	x			x	The HTTP EP adapter cannot emit events in a recoverable way and therefore does not support synchronous, transactional event emission.
		x	x		
		x		x	
TDQ	x		x		For synchronous event emission, the TD queue must be a recoverable intra-partition queue for transactional events and either an unrecoverable intra-partition queue or an extra-partition queue for non-transactional events. The TDQ EP adapter can be used for testing and debugging as well as production.
	x			x	
		x	x		
		x		x	
Transaction start		x	x		The transaction start EP adapter does not support assured emission of events.
		x		x	

Table 6. EP adapter emission and transactional modes (continued)

Event processing adapter	Emission mode		Transactional mode		Notes
	SYNC	ASYNC	TRANS	NONTRANS	
TSQ	x		x		For synchronous event emission, the TS queue must be recoverable for transactional events and unrecoverable for non-transactional events. The TSQ EP adapter is generally used for testing and debugging, not production.
	x			x	
		x	x		
		x		x	
Custom	Each combination of adapter, emission mode, and transactional mode is possible depending on how you have configured your custom EP adapter.				The custom EP adapter must conform to the recoverability requirements of the event. Consult the documentation for your custom EP adapter.

IBM MQ EP adapter

The IBM MQ EP adapter emits events to a queue in IBM MQ either in one of the XML formats or in binary format.

IBM MQ on z/OS is a prerequisite for using and implementing the IBM MQ EP adapter. To use the IBM MQ EP adapter, you must set up the CICS-IBM MQ adapter to provide a connection between the CICS region and IBM MQ on z/OS. For more information, see Setting up the CICS-MQ adapter.

The IBM MQ EP adapter places CICS events on the queue that is specified in the EP Adapter specification; the events are placed on the queue in these formats:

- “Common base event format” on page 75, to be consumed by IBM Business Monitor.
- “CICS flattened event (CFE) format” on page 78, to be consumed by application programs.
- “Decision Server Insights Event format” on page 77, to be consumed by the Decision Server Insights component of IBM Operational Decision Manager.
- “WebSphere Business Events format” on page 78, to be consumed by the Decision Server Events component of IBM Operational Decision Manager.

The IBM MQ EP adapter supports these emission and transactional modes. The letter "x" indicates a supported combination of modes.

Table 7. IBM MQ EP adapter emission and transactional modes

Event processing adapter	Emission mode		Transactional mode		Notes
	SYNC	ASYNC	TRANS	NONTRANS	
IBM MQ	x		x		Event emission is assured by using synchronous emission mode. For the IBM MQ EP adapter, event delivery is assured when synchronous emission mode is used in combination with persistent queues.
	x			x	
		x	x		
		x		x	

The event binding specification provides the queue details to the EP adapter.

Table 8. IBM MQ EP adapter properties	
Data element	**Description**
Queue name	The name of the queue where the message is placed. This element is mandatory; there is no default. For more information, see MQOD object descriptor.
Persistent	Whether the messages are persistent in the IBM MQ infrastructure. The available options are YES, NO, or QUEUE DEFAULT. The last option is the default setting, defining that the persistence of the queue is to be used. For more information, see MQMD persistence field.
Priority	The priority of the message in the IBM MQ infrastructure; the default is no priority specification with a valid range of 0 to 99999999. For more information, see MQMD priority field.
Expiry time	The expiry time in milliseconds. For more information, see MQMD expiry field.
Format	The event format: CICS Flattened Event, Common Base Event, Decision Server Insights, or WebSphere Business Events.

You specify business events and the CICS EP adapters that they use by using the Event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

When using IBM MQ as the transport for either the WebSphere Business Event or common base event form of the event, the message is placed on the message queue as UTF-8 with escape sequences for the restricted characters:

Table 9. Restricted characters and their escape sequences	
Character	**Escape sequence**
&	&amp
<	&lt
>	&gt
"	&quot
'	&apos

HTTP EP adapter

The HTTP EP adapter emits events to an HTTP 1.1 compliant server by using HTTP POST in XML format for consumption by products such as IBM Operational Decision Manager and IBM Business Monitor without using IBM MQ as your transport.

IBM MQ remains the primary method for emitting events from CICS; the HTTP EP adapter can be used as a less robust alternative. You can also use the HTTP EP adapter to send your XML format events to vendor products or any HTTP 1.1 compliant server.

To use the HTTP EP adapter, you must enable TCP/IP in your CICS region; you can enable TCP/IP in the CICS region by specifying **TCPIP=YES** in your system initialization parameters.

The HTTP EP adapter supports the following formats:

- The “Common base event format” on page 75, which is consumable by any product supporting the Common Event Infrastructure.
- The “Common base event REST format” on page 76 (the common base event format without the common base event envelope) which is consumable by IBM Business Monitor when using the IBM Business Monitor event emitter REST interface to receive events using HTTP.
- The “Decision Server Insights Event format” on page 77, to be consumed by the Decision Server Insights component of IBM Operational Decision Manager.
- The “WebSphere Business Events format” on page 78 recognized by the Decision Server Events component of IBM Operational Decision Manager.

The HTTP EP adapter converts the data into one of the supported XML formats and the converted data is then sent to an HTTP 1.1 compliant server using the connection that is described in a URIMAP.

IBM Operational Decision Manager version 7 or later and IBM Business Monitor version 7 or later are prerequisites for the HTTP transports. These prerequisites apply only when using IBM Operational Decision Manager or IBM Business Monitor as your event consumer.

The HTTP EP adapter supports these emission and transactional modes. The letter "x" indicates a supported combination of modes.

Table 10. HTTP EP adapter emission and transactional modes

Event processing adapter	Emission mode		Transactional mode		Notes
	SYNC	ASYNC	TRANS	NONTRANS	
HTTP	x			x	The HTTP EP adapter cannot emit events in a recoverable way and therefore does not support synchronous, transactional event emission.
		x	x		
		x		x	

To configure the HTTP EP adapter, you must provide two pieces of information: the name of the URIMAP that represents the connection to the HTTP 1.1 compliant server and the XML format type that is used for sending the event data.

Important: When using the HTTP EP adapter, you must specify a URIMAP with USAGE(CLIENT) in your URIMAP definition.

- For more information about URIMAP definitions and HTTP client requests from a CICS application, see URIMAP resources.
- For more information about setting up your URIMAP with IBM Operational Decision Manager, see “Setting up the Decision Server Events component of IBM Operational Decision Manager (ODM) for the HTTP EP adapter” on page 62.
- For more information about setting up your URIMAP with IBM Business Monitor, see “Setting up IBM Business Monitor for the HTTP EP adapter” on page 64.
- For more information about setting up your URIMAP with any other HTTP server, see Creating a URIMAP resource for CICS as a HTTP client.

Table 11. HTTP EP adapter properties

Data element	Description
URIMAP	The name of the URIMAP that represents the connection to the server; for example, *MYURIMAP*.
FORMAT	The event format: Common Base Event, Common Base Event REST, Decision Server Insights, or WebSphere Business Events.

You specify business events and the CICS EP adapters that they use by using the Event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

When you use the HTTP EP adapter, the Event binding editor ensures that the correct schema version is used. For more information about how schema versions are identified and incremented, see Event processing schema versions in the CICS Explorer product documentation.

The DFHECEAH program, CEPH default transaction, and DFHECEPH default transaction profile are defined for you by CICS. For more information about tuning the transaction and profile definition for performance, see Improving event processing performance.

Tip: If your target system requires basic authentication or security policies when using the HTTP EP adapter, you must implement XWBSNDO and XWBAUTH user exits to provide the required credentials. See HTTP client send exit XWBSNDO and HTTP client send exit XWBAUTH.

Setting up the Decision Server Events component of IBM Operational Decision Manager (ODM) for the HTTP EP adapter

To send events over HTTP to Decision Server Events, you must create resources in both CICS and Decision Server Events. Use these examples to help you set up and configure both environments.

Decision Server Events project configuration

The following high-level procedure shows the steps you must perform when you build your Decision Server Events project. For more information about IBM Operational Decision Manager and Decision Server Events setup, see IBM Operational Decision Manager.

1. Create an Event project using the Event Designer tool.
2. Insert a new folder.
3. Insert a new event using the template exported from the Event Binding Editor.
4. Amend the properties of the event and navigate to the **Connector** tab.
5. Select **HTTP connection**.
6. Select your preferred transport; either HTTP or HTTPS, as required.

For more information about event processing and CICS, see IBM Redbooks: Event Processing with CICS.

CICS setup for non-SSL authentication

Create a URIMAP resource with the following attributes:

```
URIMAP(mapname) GROUP(groupname)
PATH(/wbe/servlet/EventConnectorServlet)
SCHEME(HTTP)
USAGE(CLIENT)
HOST(hostname)
PORT(portnumber)
SOCKETCLOSE(hhmmss)
```

- *mapname* is the URIMAP name referred to in the event binding file.
- *groupname* is the CSD group to which the resource is to be added.
- *hostname* is the host name or IP address of Decision Server Events.
- *portnumber* is the WC_defaulthost port of Decision Server Events; the default port is 9080.
- *hhmmss* is a period of time for which CICS keeps client connections that were opened by the HTTP EP adapter available in a pool for reuse. Choose a suitable time period depending on the frequency of event emissions. If you do not set this attribute, the connection is closed after each event is emitted.

CICS setup for SSL authentication

Create a URIMAP resource with the following attributes:

```
URIMAP(mapname) GROUP(groupname)
PATH(/wbe/servlet/EventConnectorServlet)
SCHEME(HTTPS)
USAGE(CLIENT)
HOST(hostname)
PORT(portnumber)
SOCKETCLOSE(hhmmss)
```

- *mapname* is the URIMAP name referred to in the event binding file.
- *groupname* is the CSD group to which the resource is to be added.
- *hostname* is the host name or IP address of Decision Server Events.
- *portnumber* is the WC_defaulthost_secure port of Decision Server Events; the default port is 9443.
- *hhmmss* is a period of time for which CICS keeps client connections that were opened by the HTTP EP adapter available in a pool for reuse. Choose a suitable time period depending on the frequency of event emissions. If you do not set this attribute, the connection is closed after each event is emitted.

Because the URIMAP resource uses the HTTPS scheme, the CICS region requires a key ring allocated to it. You must add the signing SSL certificate of Decision Server Events to this key ring as a certificate authority before sending the event because CICS checks the server certificate for authenticity. If you add the certificate to the key ring while CICS is running, issue the PERFORM SSL REBUILD command to refresh the cache of certificates in the SSL environment for the CICS region.

Setting up the Decision Server Insights component of IBM Operational Decision Manager (ODM) for the HTTP EP adapter

To send events over HTTP to Decision Server Insights, you must create resources in both CICS and Decision Server Insights. Use these examples to help you set up and configure both environments.

Decision Server Insights project configuration

The following high-level procedure shows the steps you must perform when you build your Decision Server Insights project. For more information about the setup of the Decision Server Insights component of IBM Operational Decision Manager, see IBM Operational Decision Manager.

1. Create a Solution Project to contain the basis of the Insights solution. This step also creates a Business Object Model (BOM) project that is used to describe the business model.
2. Import the schema, as exported from CICS into the BOM project. This defines the type of events to be consumed by the Solution.
3. In the BOM Project, create a Business Model Definition file to define the entities required by the Solution.
4. Create a Rule Agent Project to contain the rules that detect situations of interest.
5. Define connectivity in the Solution Project for the events that are emitted from CICS.
6. Deploy the Solution to a running Insights Server to start receiving, and processing, events from CICS.

CICS setup for non-SSL authentication

Create a URIMAP resource with the following attributes:

```
URIMAP(mapname) GROUP(groupname)
PATH(/wbe/servlet/EventConnectorServlet)
SCHEME(HTTP)
USAGE(CLIENT)
HOST(hostname)
PORT(portnumber)
SOCKETCLOSE(hhmmss)
```

- *mapname* is the URIMAP name referred to in the event binding file.
- *groupname* is the CSD group to which the resource is to be added.
- *hostname* is the host name or IP address of Decision Server Insights.

- *portnumber* is the HTTPS port of Decision Server Insights. This can be found from the `bootstrap.properties` file of the Insight® Server.
- *hhmmss* is a period of time for which CICS keeps client connections that were opened by the HTTP EP adapter available in a pool for reuse. Choose a suitable time period depending on the frequency of event emissions. If you do not set this attribute, the connection is closed after each event is emitted.

CICS setup for SSL authentication

Create a URIMAP resource with the following attributes:

```
URIMAP(mapname) GROUP(groupname)
PATH(/wbe/servlet/EventConnectorServlet)
SCHEME(HTTPS)
USAGE(CLIENT)
HOST(hostname)
PORT(portnumber)
SOCKETCLOSE(hhmmss)
```

- *mapname* is the URIMAP name referred to in the event binding file.
- *groupname* is the CSD group to which the resource is to be added.
- *hostname* is the host name or IP address of Decision Server Insights.
- *portnumber* is the HTTPS port of Decision Server Insights. This can be found from the `bootstrap.properties` file of the Insight Server.
- *hhmmss* is a period of time for which CICS keeps client connections that were opened by the HTTP EP adapter available in a pool for reuse. Choose a suitable time period depending on the frequency of event emissions. If you do not set this attribute, the connection is closed after each event is emitted.

Because the URIMAP resource uses the HTTPS scheme, the CICS region requires a key ring allocated to it. You must add the signing SSL certificate of Decision Server Insights to this key ring as a certificate authority before sending the event because CICS checks the server certificate for authenticity. If you add the certificate to the key ring while CICS is running, issue the PERFORM SSL REBUILD command to refresh the cache of certificates in the SSL environment for the CICS region.

Setting up IBM Business Monitor for the HTTP EP adapter

To send events from CICS to a IBM Business Monitor server, you must create a URIMAP resource and specify the configuration settings you require. Use these examples to help you set up and configure your URIMAP resource.

IBM Business Monitor setup

To configure IBM Business Monitor to receive your events from CICS you must create a *Monitor Model*. For more information about IBM Business Monitor, see IBM Business Monitor.

- For more information about setting up and using IBM Business Monitor, see IBM Redbooks: Business Activity Monitoring with WebSphere Business Monitor V5.6
- For more information about event processing and CICS, see IBM Redbooks: Event Processing with CICS.

CICS setup for non-SSL with no authentication

Create a URIMAP resource with the following attributes:

```
URIMAP(mapname) GROUP(groupname)
                PATH(/rest/bpm/events)
                SCHEME(HTTP)
                USAGE(CLIENT)
                HOST(hostname)
                PORT(portnumber)
                SOCKETCLOSE(hhmmss)
```

- *mapname* is the URIMAP name referred to in the event binding file.

- *groupname* is the CSD group to which the resource is to be added.
- *hostname* is the host name or IP address of the IBM Business Monitor server.
- *portnumber* is the WC_defaulthost port of the IBM Business Monitor server; the default port is 9080.
- *hhmmss* is a period of time for which CICS keeps client connections that were opened by the HTTP EP adapter available in a pool for reuse. Choose a suitable time period depending on the frequency of event emissions. If you do not set this attribute, the connection is closed after each event is emitted.

Note: You cannot use Basic Authentication with the HTTP Scheme because the IBM Business Monitor server redirects basic authentication requests to the HTTPS port.

CICS setup for SSL with no authentication

Create a URIMAP resource with the following attributes:

```
URIMAP(mapname) GROUP(groupname)
                PATH(/rest/bpm/events)
                SCHEME(HTTPS)
                USAGE(CLIENT)
                HOST(hostname)
                PORT(portnumber)
                SOCKETCLOSE(hhmmss)
```

- *mapname* is the URIMAP name referred to in the event binding file.
- *groupname* is the CSD group to which the resource is to be added.
- *hostname* is the host name or IP address of the IBM Business Monitor server.
- *portnumber* is the WC_defaulthost_secure port of the IBM Business Monitor server; the default port is 9443.
- *hhmmss* is a period of time for which CICS keeps client connections that were opened by the HTTP EP adapter available in a pool for reuse. Choose a suitable time period depending on the frequency of event emissions. If you do not set this attribute, the connection is closed after each event is emitted.

Because the URIMAP resource uses the HTTPS scheme, the CICS region requires a key ring allocated to it. You must add the signing SSL certificate of the IBM Business Monitor server to this key ring as a certificate authority before sending the event because CICS checks the server certificate for authenticity. If you add the certificate to the key ring while the CICS region is running, issue the PERFORM SSL REBUILD command to refresh the cache of certificates in the SSL environment for the CICS region.

CICS setup for SSL with basic authentication

Create a URIMAP resource with the following attributes:

```
URIMAP(mapname) GROUP(groupname)
                PATH(/rest/bpm/events)
                SCHEME(HTTPS)
                USAGE(CLIENT)
                HOST(hostname)
                PORT(portnumber)
                AUTHENTICATE(BASIC)
                SOCKETCLOSE(hhmmss)
```

- *mapname* is the URIMAP name referred to in the event binding file.
- *groupname* is the CSD group to which the resource is to be added.
- *hostname* is the host name or IP address of the IBM Business Monitor server.
- *portnumber* is the WC_defaulthost_secure port of the IBM Business Monitor server; the default port is 9443.
- *hhmmss* is a period of time for which CICS keeps client connections that were opened by the HTTP EP adapter available in a pool for reuse. Choose a suitable time period depending on the frequency of event emissions. If you do not set this attribute, the connection is closed after each event is emitted.

Because the URIMAP resource uses the HTTPS scheme, the CICS region requires a key ring allocated to it. You must add the signing SSL certificate of the IBM Business Monitor server to this key ring as a certificate authority before sending the event because CICS checks the server certificate for authenticity. If you add the certificate to the key ring while the CICS region is running, issue the PERFORM SSL REBUILD command to refresh the cache of certificates in the SSL environment for the CICS region.

The XWBAUTH global user exit must be provided, as it is this exit that supplies the user ID and password.

Transaction start EP adapter

The transaction start EP adapter extends application behavior in an event-driven way by driving a new CICS transaction, which uses the information passed to it in the container-based event format. The CICS transaction can run on a CICS system different from the CICS system where the event occurred.

This adapter format can also be used to drive an existing CICS program, which does not need any of the event data, but can be triggered by an event occurrence.

The transaction start EP adapter is provided to call a CICS transaction on the outcome of an event. The transaction is started by the EP adapter and passes the CICS event object as input.

The transaction start EP adapter supports these emission and transactional modes. The letter "x" indicates a supported combination of modes.

Table 12. Transaction start EP adapter emission and transactional modes

Event processing adapter	Emission mode		Transactional mode		
	SYNC	ASYNC	TRANS	NONTRANS	
Transaction start		x	x		The transaction start EP adapter does not support assured emission of events.
		x		x	

Note: The transaction start EP adapter cannot emit events in a recoverable way and therefore does not support synchronous, transactional event emission.

Table 13. Transaction start EP adapter properties

Data element	Description
Transaction ID	The identifier for the transaction that is started. This identifier is mandatory and there is no default.
SYSID	The system ID of the CICS system where the transaction is to be started. The default is to leave this field blank for the local system ID.
UserID	The user ID under which the transaction is started. The default is the CICS default user ID.

The transaction start EP adapter properties apply to the transaction that is started, and not to the EP adapter itself. If you specify a user ID, the transaction runs with this user ID. If you select the option `Use Context User Id`, the transaction runs under the user ID that captured the event. If neither of them are specified, the transaction runs under the CICS default user ID.

If you install a transaction start EP adapter and specify a transaction user ID without selecting the option `Use Context User Id`, CICS checks that the installation user ID is authorized as a surrogate user of the transaction user ID. For more information, see Where surrogate user checking applies and RACF definitions for surrogate user checking.

Important: If you select a transaction start adapter for emitting a TASK threshold or TRANCLASS TASK threshold system event, and also select the option `Use Context User Id`, the context user ID is the CICS region user ID. In this instance, you might want to specify a specific user ID that you want the transaction to use when running.

The transaction start EP adapter calls the transaction as specified in the transaction ID, by **START TRANID**, passing a channel that contains the formatted data. The transaction and program must be defined to be created by the client or independent software vendors. The data is then passed, unmodified, to the transaction. The transaction start EP adapter sends the event to the started transaction in the CICS container-based event (CCE) format, where the event name values are passed in containers in a channel DFHEP.EVENT.

Note: If the **START TRANSID** refers to a remote transaction, security for the CICS region where the started task executes needs to allow function shipping of the **START TRANSID** command. For example with MRO, the FMH-5 that is used when the **START TRANSID** command is shipped contains the CICS region user ID of the CICS region that emitted the event. MRO security for the CICS region where the started task executes may need to allow for this CICS region user ID being passed in the FMH-5. For more information about security when a remote transaction is used, see Intercommunication security.

You specify business events and the CICS EP adapters that they use by using the Event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

Transient data queue (TDQ) EP adapter

You can use the transient data queue (TDQ) EP adapter to emit events to any consumer that can read the event from the TD queue.

You also can use this EP adapter for the following tasks:

- Processing event data from transient data records.
- Developing and testing event specifications.
- Testing whether events are captured when expected and whether they contain the expected data items.

The TDQ EP adapter puts CICS events on the transient data queue that is specified in the EP adapter specification.

The TDQ EP adapter supports the following formats:

- “Common base event format” on page 75
- “Common base event REST format” on page 76
- “CICS flattened event (CFE) format” on page 78
- “Decision Server Insights Event format” on page 77
- “WebSphere Business Events format” on page 78

The following table shows the emission and transactional modes that the TDQ EP adapter supports. The letter "x" indicates a supported combination of modes.

Table 14. TDQ EP adapter emission and transactional modes

Event processing adapter	Emission mode		Transactional mode		Notes
	SYNC	ASYNC	TRANS	NONTRANS	
TDQ	x		x		For synchronous event emission, the TD queue must be a recoverable intra-partition queue for transactional events and either an unrecoverable intra-partition queue or an extra-partition queue for non-transactional events. The TDQ EP adapter can be used for testing and debugging as well as production.
	x			x	
		x	x		
		x		x	

The following table shows the properties of the TDQ EP adapter.

Table 15. Transient data queue (TDQ) EP adapter properties	
Data Element	**Description**
Queue name	The name of the TD queue to which you want the TDQ EP adapter to send events. If the named TD queue is extra-partition, it must be defined with TYPEFILE(OUTPUT). For synchronous event emission, the TD queue must be a recoverable intra-partition transient data queue that is defined with RECOVSTATUS(LOGICAL) or RECOVSTATUS(PHYSICAL) for transactional events.
CICS system ID | blank (default)	The CICS system ID of the TD queue to which the event is to be written. The default is to leave this field blank for the local system ID.

You specify business events and the CICS EP adapters that they use by using the Event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

The DFHECEAQ program name and CEPR default transaction are defined for you by CICS.

A logical event contains context and formatted captured data values. The logical event is written to a single transient data queue record.

The event data comes from containers each with a maximum data length of 2097152 KB. Transient data queues support 32763 bytes of data per record; events that exceed this record length are truncated. The TDQ EP adapter supports 4-byte queue names and writes 1 TD queue record per event.

Temporary storage queue (TSQ) EP adapter

You can use the temporary storage queue (TSQ) EP adapter to emit events to any consumer that can read the event from the TS queue.

You also can use this EP adapter for the following tasks:

- Processing event data from temporary storage records.
- Developing and testing event specifications.
- Testing whether events are captured when expected and whether they contain the expected data items.

The TSQ EP adapter puts CICS events on the temporary storage queue that is specified in the EP adapter specification.

The TSQ EP adapter supports the following formats:

- “Common base event format” on page 75
- “Common base event REST format” on page 76
- “CICS flattened event (CFE) format” on page 78
- “Decision Server Insights Event format” on page 77
- “WebSphere Business Events format” on page 78

The following table shows the emission and transactional modes that the TSQ EP adapter supports. The letter "x" indicates a supported combination of modes.

Table 16. TSQ EP adapter emission and transactional modes					
Event processing adapter	**Emission mode**		**Transactional mode**		**Notes**
	SYNC	**ASYNC**	**TRANS**	**NONTRANS**	
TSQ	x		x		For synchronous event emission, the TS queue must be recoverable for transactional events and unrecoverable for non-transactional events. The TSQ EP adapter is generally used for testing and debugging, not production.
	x			x	
		x	x		
		x		x	

The following table shows the properties of the TSQ EP adapter.

Table 17. Temporary storage queue (TSQ) EP adapter properties	
Data Element	**Description**
Queue name	The name of the TS queue to which you want the TSQ EP adapter to send events.
CICS system ID | blank (default)	The CICS system ID of the TS queue to which the event is to be written. The default is to leave this field blank for the local system ID.
Use auxiliary temporary storage	A check box to select the use of auxiliary temporary storage.

You specify business events and the CICS EP adapters that they use by using the Event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

The DFHECEAT program name and CEPT default transaction are defined for you by CICS.

A logical event contains context and formatted captured data values. The logical event is written to a single temporary storage queue record.

Main temporary storage is typically used if small amounts of data are to be stored, the data is needed for only short periods of time, and the data does not need to be recoverable. Main temporary storage data is in 64-bit (above-the-bar) storage.

Auxiliary temporary storage is typically used if large volumes of data are to be stored, the data is needed for extended periods of time, and the data is to be maintained from one CICS run to the next. Auxiliary temporary storage queue data is held on the CICS-managed VSAM Entry-Sequenced Data Set (ESDS) DFHTEMP and uses 31-bit storage. It can be defined as a recoverable CICS resource.

The queue can be defined as recoverable because event processing does not delete the queue record after writing it.

If you use a temporary data set when writing the temporary storage records and the data set becomes full, CICS suspends the EP adapter until space becomes available.

The TSQ EP adapter writes the TS queue record with the NOSUSPEND option to prevent the suspension of either the dispatcher task or the adapter task when a user ID is specified for the adapter task.

If the default value of AUXILIARY is used, data is written to DFHTEMP. When MAIN is specified, data is written to main temporary storage. Either value can be overridden by a TSMODEL or temporary storage pool.

The event data comes from containers each with a maximum data length of 2097152 KB. Temporary storage queues support 32763 bytes of data per record; events that exceed this record length are truncated. The TSQ EP adapter supports 16-byte queue names and writes 1 TS queue record per event.

Custom EP adapter

You can write a custom EP adapter if the CICS-supplied EP adapters do not meet your needs.

A custom EP adapter is a CICS program you write to format and route an event. A custom EP adapter might be needed when one of the CICS-supplied EP adapters does not fulfil your needs. A custom EP adapter should not carry out any other processing, such as consuming the event.

When a custom EP adapter is processing asynchronous events, it is started by attaching a transaction. The transaction must be a local transaction and is not eligible for static or dynamic routing. When a custom EP adapter is processing synchronous (assured) events, it is linked to as a user-replaceable program; it is started by program LINK and executes in CICS key regardless of the EXECKEY option specified in its program resource definition. The EXECKEY option is honored when a custom EP adapter is attached.

The custom EP adapter supports these emission and transactional modes:

Table 18. Custom EP adapter emission and transactional modes

Event processing adapter	Emission mode		Transactional mode		Notes
	SYNC	ASYNC	TRANS	NONTRANS	
Custom	Each combination of adapter, emission mode, and transactional mode is possible depending on how you have configured your custom EP adapter.				The custom EP adapter must conform to the recoverability requirements of the event. Consult the documentation for your custom EP adapter.

You specify business events and the CICS EP adapters that they use by using the Event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

CICS supports custom EP adapters written in the following languages:

COBOL
PL/I
C
C++
Assembler language.

Writing a custom EP adapter

A custom event processing (EP) adapter is a CICS program associated with an event binding that formats and then emits events produced by the event binding.

About this task

If the CICS-supplied EP adapters do not meet your requirements for processing events, you can write your own custom EP adapter to process the event data yourself.

CICS invokes the EP adapter for each event that is emitted. The input to a custom EP adapter is the current channel, which contains the CICS event object as a collection of containers. The containers are: `DFHEP.CONTEXT`, `DFHEP.DESCRIPTOR`, `DFHEP.ADAPTER`, `DFHEP.ADAPTPARM`, `DFHEP.CHAR.`*nnnnn*, and `DFHEP.DATA.`*nnnnn*. Copybooks are provided for the `DFHEP.CONTEXT`, `DFHEP.DESCRIPTOR`, and `DFHEP.ADAPTPARM` containers. These copybooks can change between releases of CICS; therefore, you should recompile custom EP adapters for each new CICS release.

In addition to the emitted event, the custom EP adapter must produce an indication of success or failure.

Procedure

1. Include the event context data copybook for your programming language.

   This copybook describes the `DFHEP.CONTEXT` container of context data for the event your EP adapter is processing.

- DFHEPCXD for Assembler language
- DFHEPCX0 for COBOL
- DFHEPCXL for PL/I
- DFHEPCXH for C

2. Include the event descriptor copybook for your programming language.

   This copybook describes the DFHEP.DESCRIPTOR container that describes the captured business data for the event your EP adapter is processing.

   - DFHEPDED for Assembler language
   - DFHEPDE0 for COBOL
   - DFHEPDEL for PL/I
   - DFHEPDEH for C

3. Include the EP adapter parameters copybook for your programming language.

   This copybook describes the DFHEP.ADAPTPARM container.

   - DFHEPAPD for Assembler language
   - DFHEPAP0 for COBOL
   - DFHEPAPL for PL/I
   - DFHEPAPH for C

4. Get the context information from the DFHEP.CONTEXT container using the event context data copybook.
5. Get the EP adapter custom data. The DFHEP.ADAPTER container contains the data that is specified in the field **Data passed to the Custom Adapter** that is in the **Adapter** tab of the event binding editor for your event binding.

   Note: The data in this container is restricted to EBCDIC characters.

6. Get the EP adapter parameters from the DFHEP.ADAPTPARM container using the EP adapter parameters configuration copy book.

   An important item of information in the DFHEP.ADAPTPARM container is the emission recoverability indicator, EPAP-RECOVER. This container also includes the adapter name, EPAP-ADAPTER-NAME.

7. The custom adapter must verify the EPAP-RECOVER setting in the DFHEP.ADAPTPARM container in the custom EP adapter.

   EP adapters must emit the events by using a transport in a recoverable or unrecoverable way. If the setting is not EPAP-ANY-RECOVERABLE, you must honor the EPAP-RECOVER setting. To support synchronous emission, EP adapters must be sensitive to the transport recoverability requirement that is indicated by the EPAP-RECOVER setting in the context container:

   - If the field is set to EPAP-RECOVERABLE, the EP adapter must write to the transport in a recoverable way.
   - If the field is set to EPAP-NON-RECOVERABLE, the EP adapter must ***not*** write to the transport in a recoverable way.

   You must terminate the adapter if you cannot honor the recoverability setting; otherwise, the transactional requirement for the event is not implemented correctly.

   The field is set to EPAP-ANY-RECOVERABLE for asynchronous emission.

8. Get the event descriptor from the DFHEP.DESCRIPTOR container using the event descriptor copybook.

   The event descriptor consists of a prefix describing the number of data items in the descriptor and a descriptor for each data item. The data item descriptor includes the name of the business data item and its type. The data items themselves are in containers with a name of format DFHEP.CHAR.*nnnnn* and DFHEP.DATA.*nnnnn*, where *nnnnn* is a 5-digit sequence number that indicates the ordering of the captured data starting at 00001. The DFHEP.CHAR.*nnnnn* containers

contain the capture data in a printable (character) form formatted as requested in the event specification. The DFHEP.DATA.*nnnnn* containers contain the unformatted capture data.

9. Get data items from the DFHEP.CHAR.*nnnnn* or DFHEP.DATA.*nnnnn* containers.

   Process the items of captured data. For example, you might want to write each item to a CICS transient data queue.

   Note: If a data item was not available for data capture, the corresponding data item container is not present in the CICS event object. This situation can happen, for example, when CAPTURESPEC specifies a capture data item associated with an optional parameter that was not present on the API command that caused the event. The corresponding DFHEP.CHAR.*nnnnn* will be present in the CICS event object but will contain one or more asterisks, one asterisk for each character of missing data.

10. Format the data and emit the event.

    Each item in the DESCRIPTOR array defines the type of the source data captured and the required length and type of that data when and if it is formatted.

    Data is captured up to the length specified in the capture data item spec. The captured data is exactly as found in the source data areas. If the capture data is not available then the corresponding DFHEP.DATA container is not created.

    A format length of **Automatic** (0) is supported for all data types. When a format length of **Automatic** is used, the equivalent EPDE field is set to `epde_formatLen_auto`.

    A format precision of **Automatic**, resulting in the length that is required for the specified data type and precision being used, is supported for all numeric data types. When a format precision of **Automatic** is used, the equivalent EPDE field is set to `epde_formatPrec_auto`.

11. Complete processing by issuing an **`EXEC CICS RETURN`** or **`EXEC CICS ABEND`** command.

    In addition to the emitted event, the custom EP adapter must produce an indication of success or failure. A custom EP adapter that cannot emit the event must end with an abend code so that CICS can start any required recovery actions and increment the appropriate statistics.

An example code fragment in the COBOL language

This code fragment shows the sequence of steps described in this procedure. It does not include any processing of the EP adapter information or the data items.

```
*****************************************************************
 Linkage section.
*****************************************************************
 01 EPContext.
    copy dfhepcxo.
 01 EPDescriptor.
    copy dfhepdeo.
 01 EPAdapter          pic x(16).
 01 EPAdaptparm
    copy dfhepapo.
 01 EPData             pic x(32000).
*****************************************************************
 Main-program section.
*****************************************************************
*
     perform Initial-processing.
*
*    Process the data items
     perform Process-data-item
             varying ItemNum from 1 by 1
             until ItemNum > epde-itemcount.
*
*****************************************************************
* Any final EVENT PROCESSING code to go here
*****************************************************************
*
*    Return to caller
     EXEC CICS RETURN END-EXEC.
*
 Main-program-exit.
     exit.
*
```

```
*****************************************************************
 Initial-processing section.
*****************************************************************
*
*    Obtain the DFHEP.CONTEXT container
     EXEC CICS GET CONTAINER('DFHEP.CONTEXT')
                   SET(address of EPContext)
                   FLENGTH(EPContextLength)
     END-EXEC.
*
*    Obtain the DFHEP.DESCRIPTOR container
     EXEC CICS GET CONTAINER('DFHEP.DESCRIPTOR')
                   SET(address of EPDescriptor)
                   FLENGTH(EPDescriptorLength)
     END-EXEC.
*
*    Obtain the DFHEP.ADAPTER container
     EXEC CICS GET CONTAINER('DFHEP.ADAPTER')
                   SET(address of EPAdapter)
                   FLENGTH(EPAdapterLength)
     END-EXEC.
*
*    Obtain the DFHEP.ADAPTPARM container
     EXEC CICS GET CONTAINER('DFHEP.ADAPTPARM')
                   SET(address of EPAdaptparm)
                   FLENGTH(EPAdaptparmLength)
     END-EXEC.
*

*    Check the recoverability of the transport is right for the event
     if not epap-any-recoverable
       perform Check-recoverability.

*
 Initial-processing-exit.
     exit.
*
*****************************************************************
 Process-data-item section.
*****************************************************************
*
*    Process a data descriptor item
*
*    Build the data container name: DFHEP.DATA.nnnnn
     string 'DFHEP.DATA.' delimited by size
            ItemNum         delimited by size
            into ContainerName
     end-string.
*
*    Obtain the DFHEP.DATA.nnnnn container - if present
     EXEC CICS GET CONTAINER(ContainerName)
                   SET(address of EPData)
                   FLENGTH(EPDataLength)
                   RESP(Resp) RESP2(Resp2)
     END-EXEC.
*****************************************************************
* Any final code to process DATA ITEM to go here
*****************************************************************
*
*    Convert the data according to epde-datatype
     perform Convert-data.
*
*    Calculate the target field length
     move epde-formatlen of epde-item(ItemNum) to TSQFieldLength
     if TSQFieldLength = 0 and Resp = dfhresp(normal)
       move TSQItemLength to TSQFieldLength
     end-if
     if 32001 - TSQFieldIndex < TSQFieldLength
       compute TSQFieldLength = 32001 - TSQFieldIndex
     end-if.
*
*    Format the data according to epde-formattype
     perform Format-data.
*
*    Move over the data item ready for the next one
     add TSQFieldLength to TSQFieldIndex.
*
 Process-data-item-exit.
     exit.
*
*****************************************************************
```

Sample custom EP adapter

To help you develop your own custom EP adapter, a sample is provided as source code and also as a load module.

The sample custom EP adapter is provided in the COBOL language. It is shipped as source code in the CICSTS56 .CICS.SDFHSAMP library, and also as a load module.

- The source code and load module are named DFH0EPAC.
- Group DFH$EPAG is defined in DFHCURDS.DATA. The group defines program DFH0EPAC and transaction ID EPAT to include in your event binding to run the DFH0EPAC program.
- The sample program DFH0EPAC formats most data types. However, as a COBOL language sample, DFH0EPAC cannot format binary floating point (BFP) or decimal floating point (DFP) items; in this case, DFH0EPAC fills the data area with asterisks (*).

The program is passed a CICS event channel. It formats the event data into an event record according to formatting information in the DFHEP.DESCRIPTOR container and writes the record to a CICS temporary storage queue (TSQ). The name of the queue can be taken from the DFHEP.ADAPTER container, and can be up to 16 bytes long. If that is not present, the name defaults to *userid*|*program* for application events or *userid*.SYSTEM for system events, constructed from the information in the DFHEP.CONTEXT container. For example, for an application event triggered by program TEST1 running using user ID JBLOGGS, the queue name defaults to *JBLOGGS.TEST1*. Records longer than 32,000 bytes are truncated.

The sample custom EP adapter demonstrates how a custom EP adapter handles synchronous and asynchronous emission events. This is achieved by honoring the EPAP-RECOVER flag setting in the DFHEP.ADAPTPARM container by checking whether the temporary storage queue is recoverable or not.

Running the sample custom EP adapter

The sample custom Event Processing (EP) adapter is provided as a model to help you create your own custom EP adapter.

Before you begin

For assured event emission, synchronous transactional events must have a recoverable temporary storage (TS) queue and synchronous nontransactional events must have an unrecoverable TS queue. Whether or not TS queues are recoverable depends on the value of a matching TSMODEL option. TS queues are recoverable only when there is a matching TSMODEL option that specifies that the TS queue is recoverable. When you use the sample custom EP adapter for synchronous transactional events, your program cannot issue a DELETEQ TS command for the event TS queue in the unit of work that captures the event.

About this task

To use the sample custom EP adapter, use the event binding editor to create and deploy an event binding. From CICS, you then make the sample custom EP adapter available.

Procedure

1. From the event binding editor **Adapter** tab, select **Custom (User Written)** in the **Adapter** field.
2. Depending on the type of event, enter the required information:
   - For asynchronous events, type EPAT in the **Transaction ID** field.
   - For synchronous events, type DFH0EPAC in the **Program ID** field.
3. In the **Data passed to the Custom Adapter** field, type the name of the TS queue to which you want the events written, or leave this field blank if you want the sample custom EP adapter to create a TS queue name from the user ID and program name of the program that the events originated from.
4. Optional: For synchronous emission, click the **Sync** radio button in the **Emission Mode** field in Advanced Options.
5. Optional: For transactional events, select the **Events are Transactional** check box in Advanced Options.

6. Complete the event binding.

   For more information, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.
7. Deploy the event binding to your CICS system.

   For more information, see Deploying a CICS bundle in the CICS Explorer product documentation.
8. On your CICS system, install the event binding in a bundle.

   For more information about installing your event binding bundle, see Enabling an event binding in the CICS Explorer product documentation.
9. Make the sample custom EP adapter available by installing group DFH$EPAG.

   For more information, see Supplied resource definitions, groups, and lists.
10. Run the program from which you want to generate events.

What to do next

Check the TS queue for results; click **Operations** > **Queues** > **TS Queues**.

Common base event format

The common base event format is based on the Common Base Event specification version 1.0.1, an XML-based format, using the `xs:any` slot in the common base event schema for the CICS event data. It can be used by any consumer that can recognize the common base event XML format.

You specify the event format you want to use by using the event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

This code fragment shows the format emitted if you select the common base event format in the Event binding editor. Fields later in this section in braces "{}" are fields that are derived from either the captured event or the CICS region on which the event was captured.

```
<?xml version="1.0"?>
<cbe:CommonBaseEvent xmlns:cbe="http://www.ibm.com/AC/commonbaseevent1_0_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.0.1"
creationTime="{yyyy-mm-ddThh:mm:ssZ}">
<cbe:sourceComponentId component="IBM CICS TS#5.6.0"
componentIdType="ProductName" executionEnvironment="IBM z/OS"
instanceId="{NETQUAL}.{APPLID}" location="{MVSSYSTEM}"
locationType="Hostname" subComponent="CICS EP"
componentType="http://www.ibm.com/xmlns/prod/cics/eventprocessing" />
<cbe:situation categoryName="OtherSituation">
<cbe:situationType xsi:type="OtherSituation"
reasoningScope="EXTERNAL">
<CICSApplicationEvent/>
</cbe:situationType>
</cbe:situation>
<cics:event
xmlns:cics="http://www.ibm.com/xmlns/prod/cics/events/CBE">
<cics:context-info>
<cics:eventname>{BusinessEvent}</cics:eventname>
<cics:usertag>{User Tag}</cics:usertag>
<cics:networkapplid>{NETQUAL}.{APPLID}</cics:networkapplid>
<cics:timestamp>yyyy-mm-ddThh:mm:ss.mmmZ</cics:timestamp> 1
<cics:bindingname>{Event Binding Name}</cics:bindingname>
<cics:capturespecname>{Capture Spec Name}</cics:capturespecname>
<cics:UOWid>{network unit of work id}</cics:UOWid>
</cics:context-info>
<cics:payload-data>
<data:payload xmlns:data="http://www.ibm.com/prod/cics/{User Tag}/{BusinessEvent}">
<data:customer>IBM</data:customer>
<data:orderValue>10250</data:orderValue>
<data:orderReference>QWERTY098765</data:orderReference>
</data:payload>
</cics:payload-data>
</cics:event>
</cbe:CommonBaseEvent>
```

Figure 8. Common base event format example

Note: 1 All events are emitted with Coordinated Universal Time (UTC) timestamps.

The `xs:any` element in a CICS common base event event format contains both a static part (<cics:event> tag), which is the same for every CICS common base event format, and a dynamic part (<data:payload> tag), which is different for each event specification. The static and dynamic parts are described by separate XML schemas.

The static XML schema is installed with CICS in the location: `/usr/lpp/cicsts/cicsts56 /schemas/eventprocessing/eventformats/cics_cbe_static.xsd`. You can export the dynamic schema for a CICS common base event event format from the **Adapter** tab in the Event binding editor; see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation for more details.

The CICS event context and payload data are contained in the <cics:context-info> and <cics:payload-data> child elements of the <cics:event> element. The dynamic schema provides the mapping for the payload data.

When sending CICS common base event format events to an event consumer that does not recognize the common base event format, for example, IBM WebSphere Message Broker, use the static XML schema for the common base event format. For more information, see Common Base Event XML schema.

Common base event REST format

The common base event REST format is a basic XML representation of a CICS event. The common base event REST format can be consumed by any event processing HTTP server that requires events in XML format.

You specify the event format you want to use by using the event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

This format is an XML-based format that represents the CICS event. The format is identical to the XML in the `xs:any` slot of the common base event format.

This code fragment shows the format emitted if you select the common base event REST format in the Event binding editor. Fields later in this section in braces "{}" are fields that are derived from either the captured event or the CICS region on which the event was captured.

```
<?xml version="1.0"?>
<cics:event
xmlns:cics="http://www.ibm.com/xmlns/prod/cics/events/CBE">
<cics:context-info>
<cics:eventname>{BusinessEvent}</cics:eventname>
<cics:usertag>{User Tag}</cics:usertag>
<cics:networkapplid>{NETQUAL}.{APPLID}</cics:networkapplid>
<cics:timestamp>yyyy-mm-ddThh:mm:ssZ</cics:timestamp> 1
<cics:bindingname>{Event Binding Name}</cics:bindingname>
<cics:capturespecname >{Capture Spec Name}</cics:capturespecname >
<cics:UOWid>{network unit of work id}</cics:UOWid>
</cics:context-info>
<cics:payload-data>
<data:payload xmlns:data="http://www.ibm.com/prod/cics/{User Tag}/{BusinessEvent}">
<data:customer>IBM</data:customer>
<data:orderValue>10250</data:orderValue>
<data:orderReference>QWERTY098765</data:orderReference>
</data:payload>
</cics:payload-data>
</cics:event>
```

Figure 9. common base event REST format example

Note: 1 All events are emitted with Coordinated Universal Time (UTC) timestamps.

The CICS common base event REST format contains both a static part (<cics:event> tag), which is the same for every CICS common base event REST event, and a dynamic part (<data:payload> tag), which is different for each event specification. The static and dynamic parts are described by separate XML schemas.

The static XML schema is installed with CICS in: `/usr/lpp/cists56 /schemas/eventprocessing/eventformats/cics_cbe_static.xsd`. You can export the dynamic schema for a common base event REST format event from the **Adapter** tab in the Event binding editor; see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation for more details.

The CICS event context and payload data are contained in the <cics:context-info> and <cics:payload-data> child elements of the <cics:event> element. The dynamic schema provides the mapping for the payload data.

Decision Server Insights Event format

The Decision Server Insights Event format is an XML representation of a CICS event that is recognized by the Decision Server Insights component of IBM Operational Decision Manager. This format can also be used by any consumer that can recognize the Decision Server Insights Event format.

You specify the event format you want to use by using the event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

This code fragment shows the format emitted if you select the Decision Server Insights Event format in the Event binding editor. Fields later in this section in braces "{}" are those derived from either the captured event or the CICS region on which the event was captured.

```
<?xml version="1.0" encoding="UTF-8" ?>
<event:{business-event-name} xmlns:event="http://www.ibm.com/prod/cics/{event-binding-user-tag}/{business-event-name}">
    <event:cics_usertag>{event-binding-user-tag}</event:cics_usertag>
    <event:cics_networkapplid>{NETQUAL}.{APPLID}</event:cics_networkapplid>
    <event:cics_timestamp>{yyyy-mm-ddThh:mm:ss.mmm+00:00}</event:cics_timestamp> 1
    <event:cics_bindingname>{event-binding-name}</event:cics_bindingname>
    <event:cics_capturespecname>{capture-specification-name}</event:cics_capturespecname>
    <event:cics_UOWid>{Network-UOWid}</event:cics_UOWid>
    <event:{business-info-name1}>{business-info-value1}</event:{business-info-name1}>
    <event:{business-info-name2}>{business-info-value2}</event:{business-info-name2}>
```

```
    <event:{business-info-name3}>{business-info-value3}</event:{business-info-name3}>
</event:{business-event-name}>
```

Note: 1 All events are emitted with Coordinated Universal Time (UTC) timestamps.

You can export the schema for a Decision Server Insights Event format event from the **Adapter** tab in the Event binding editor. See Specifying EP adapter and dispatcher information in the CICS Explorer product documentation for more details.

WebSphere Business Events format

The WebSphere Business Events (XML) format is based on XML in the version 2.2 format recognized by the Decision Server Events component of IBM Operational Decision Manager. This format can also be used by any consumer that can recognize the WebSphere Business Events (XML) format.

You specify the event format you want to use by using the event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

This code fragment shows the format emitted if you select WebSphere Business Events (XML) format in the Event binding editor. Fields later in this section in braces "{}" are those derived from either the captured event or the CICS region on which the event was captured.

```
<?xml version="1.0" standalone="yes"?>
<connector name="{event-binding-name}" version="2.2">
<connector-bundle name="{business-event-name}" type="Event">
<connector-object name="{business-event-name} Context">
<field type="String" name="Binding user tag">{event binding user tag}</field>
<field type="String" name="Network UOWID">{Network UOWid}</field>
</connector-object>
<connector-object name="{business-event-name} Data">
<field name="{business-info-name1}">{business-info-value1}</field>
<field name="{business-info-name2}">{business-info-value2}</field>
<field name="{business-info-name3}">{business-info-value3}</field>
</connector-object>
</connector-bundle>
<system>{NETQUAL}.{APPLID}</system>
<timestamp>{yyyy-mm-ddThh:mm:ss.mmm+00:00}</timestamp> 1
</connector>
```

Note: 1 All events are emitted with Coordinated Universal Time (UTC) timestamps.

You can export the schema for a WebSphere Business Events (XML) format event from the **Adapter** tab in the Event binding editor. See Specifying EP adapter and dispatcher information in the CICS Explorer product documentation for more details.

CICS flattened event (CFE) format

The CICS Flattened Event (CFE) format option specified assembles the event context and data values as a programming data structure.

You specify the event format you want to use by using the event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

The CFE format has a static portion containing context data and a dynamic portion containing the data specific to each event.

The data can be consumed by using one of the mapping structures shown in the following table, in conjunction with a layout of the named event defined in the event binding editor:

Table 19. Mapping structures

Name	Language	Library
DFHEPFED	Assembler	SDFHMAC

Table 19. Mapping structures (continued)		
Name	**Language**	**Library**
DFHEPFEH	C/C++	SDFHC370
DFHEPFEL	PL/I	SDFHPL1
DFHEPFE0	COBOL	SDFHCOB

The EPFE structure consists of contextual data associated with the event. For example:

- EPFE
  - EPFE_Context_Data
    - The event context identifier.
    - The event context version.
    - The event context event binding name.
    - The event binding user tag.
    - The event business name.
    - The network unit of work ID.[1]
    - The network qualified applid.
    - The capture date and time.[2]
    - The capture specification name.

Note:

[1]The network unit of work ID is set to binary zeros (NULL) for a policy event or system event.

[2] All events are emitted with Coordinated Universal Time (UTC) timestamps.

The dynamic portion of the event follows `EPFE_Context_Data` in a section called `EPFEEventData`. This data is in a non-XML, text-based format. The length and order of each item in the `EPFEEventData` section is as defined in the event specification. The length and order of the items is specified in the **Emitted Business Information** section in the event binding editor.

You can use the event binding editor to export a COBOL copybook that describes the `EPFEEventData` portion of the CFE event. If you have an event binding that was not created by using the event binding editor, you can load it into the event binding editor to export the corresponding copybook, or to get the expected contents of the event from the `EventInformationItem` details in the event binding.

When a format length is specified as **Automatic** in the Emitted Business Information table in the event binding editor, this indicates that the length of the emitted business information item is derived from the length and type of the captured field. (When there is more than one capture specification, the first one is used.) This is useful for avoiding the situation where the specified length is too small, which results in the data being truncated or emitted as asterisks.

Another option, the capture length specified for a data item in the capture specification **Information Sources**, is where a length of 0 means capture up to the end of the data area or container. This is useful when emitting, for example, the contents of a container regardless of its length.

When the event binding editor cannot determine the length required for the emitted business information, an error window opens on the copybook export and a copybook cannot be created. This is the case, for example, when the format length for a CHAR or HEX item is **Automatic** and the capture length is 0. You should specify a nonzero length for the emitted data item; otherwise, any application that consumes the event needs to know how to interpret the data items. Similarly, when floating point data is formatted as numeric or text, the length of the emitted business information depends on the value of the data field and is not known until run time. You should specify a nonzero length and possibly also a precision value for the emitted business information item.

When null-terminated strings are captured, the formatted event data is not null-terminated, so that the processing program can use the correct function for character arrays that are not null-terminated. The data types in DFHEPDE are CHARZ or HEXZ (rather than CHAR or HEX) so that the EP adapter can create the null-terminated item again, if required.

Truncated data is handled in the following ways:

- Numeric data where the size exceeds the value of the **Format length** field is emitted as asterisks, to indicate that an overflow has occurred. The **Format length** is the length specified for the **Emitted business information** field in the event binding editor.
- Text data where the size exceeds the value of the **Format length** field is truncated.
- Hex data which is of the type **Hexadecimal** in the capture specification and which exceeds the value of the **Format length** field is truncated if the format type is **Character**, or it is emitted as asterisks if the format type is numeric.

CICS flattened event (CFE) data formats

When you specify an event binding using an EP adapter with the CICS flattened event format, the data items in your event are emitted as a single event record. The event processing support in CICS processes the data in an event specification according to a set of rules.

You can use the Event binding editor to export a COBOL copybook that describes the event structure if you want to understand the data formats used in your event.

Exporting an event specification from the Event binding editor uses the same rules to create the exported copybook, so you would not normally need to understand these rules. However, understanding the rules can help you to identify the source of data formatting errors that can affect event processing. The format of an individual data item is based on the type, length, and precision you specify when defining an emitted business information item in an event specification, and the type, length and precision you specify when defining the corresponding information source in the capture specification.

- When the emitted business information is formatted as **text** the information is left-aligned and blank padded.
- When the emitted business information is formatted as **numeric** the information is right-aligned and leading zeros are added as required.
- When the emitted business information is formatted as **scientific** a number is left-aligned and blank padded in normalized E format with the (**formatPrecision**) specified number of digits after the decimal point in the mantissa. The format is `±d[.d...]E±ddd...` where `[d...]` represents one or more digits; for example, `+1E+oo3, −1.25E+000, +9.999999E−316, +0E+000`. If the specified format length is not enough for the data formatted as numeric or scientific data, number, the output area is filled with asterisks (*); text data is truncated..

Floating point data that is formatted as **text** or **numeric** has the following characteristics:

- Any nonzero precision must be less than or equal to the format length – 3 to allow for a sign, at least one leading digit, and a decimal point.
- Data with a format length of **Automatic** contains all significant whole (integer) digits.
- Data with a format precision of **Automatic** contains all significant fractional digits, unless this is precluded by an explicit format length.

Floating point data that is formatted as **scientific** has the following characteristics:

- CICS EP adapters enforce a maximum format precision (number of mantissa digits after the decimal point) of 6 for short and 15 for long floating point numbers.
- Data with a format precision of **Automatic** and a format length of **Automatic** contains all significant fractional digits.

  Padded as necessary to the maximum size for the data captured, short floating point results in a 14-byte character string (+n.nnnnnnE+nnn) and long floating point results in a 23-byte string (+n.nnnnnnnnnnnnnnE+nnn).

- Data with a format precision of **Automatic** and an explicit format length could have the precision reduced or the data padded on the right with blanks. In this case, the precision must be ≤ format length −8.
- Data with a format length of **Automatic** and an explicit format precision that is greater than 0 has a fixed length based on the format precision plus 8.

Filter and capture precision are meaningless for floating point data types and are ignored at event binding installation if they are set.

When the number is +∞, ∞, or NaN (not a number), the output area is formatted as INF, −INF, and NaN.

Note: When a mantissa is shortened to fit the specified formatPrecision, there is no rounding. However, rounding might occur in the conversion from hexadecimal floating point (HFP) or binary floating point (BFP) to decimal.

Information source field with capture type character

Emitted Business Information		**Information Source**	**COBOL Copybook**		**Comments**
Type	**Length**	**Length**	**Length**	**Format**	
text	r	*n*	r	X(r)	• If r<*n*, the field is truncated. • If r>*n*, the field is blank padded.
numeric	r	*n*	r	X(r)	• If r<*n* the field is set to all asterisks. • If r>*n* the field is blank padded.
text or numeric	0	*n*	n	X(*n*)	When exporting an event specification, *n* is taken from the first capture specification. When CICS is processing a capture specification with a field of this type, *n* is taken from the current capture specification.

Emitted Business Information		Information Source	COBOL Copybook		Comments
Type	**Length**	**Length**	**Length**	**Format**	
text or numeric	0	0			You cannot export an event specification with a field of this type with a length of zero. When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted.

Information source field with capture type hexadecimal

Emitted Business Information		Information Source	COBOL Copybook		Comments
Type	**Length**	**Length**	**Length**	**Format**	
text	r	*n*	r	"0x"||X(r-2)	• If r<(*n**2)+2, the field is truncated. • If r>(*n**2)+2 the field is blank padded.
numeric	r	*n*	r	"0x"||X(r-2)	• If r<(*n**2)+2 the field is set to all asterisks. • If r>(*n**2)+2 the field is zero padded.

Emitted Business Information		Information Source	COBOL Copybook		Comments
Type	**Length**	**Length**	**Length**	**Format**	
text or numeric	0	*n*	2+(*n**2)	"0x"||X(*n**2)	When exporting an event specification, *n* is taken from the first capture specification. When CICS is processing a capture specification with a field of this type, *n* is taken from the current capture specification.
text or numeric	0	0			You cannot export an event specification with a field of this type with a length of 0. When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted.

Information source field with capture type halfword

The table applies to information source fields with Capture Type Unsigned Halfword and Signed Halfword.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	r	0	2	r	+9(r-1)	• If r-1<digits, the field is set to all asterisks. • If r-1>digits, the field is zero padded.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	r	s	2	r	+9(r-s-2).9(s)	• If r-s-2<digits are required, the field is set to all asterisks. • If r-s-2>digits are required, the field is zero padded.
numeric	Automatic	0	2	6	+9(5)	
numeric	Automatic	s	2	7+s	+9(5).9(s)	

Information source field with capture type fullword

The table applies to information source fields with capture type unsigned and signed fullword.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	r	0	4	r	+9(r-1)	• If r-1<digits are required, the field is set to all asterisks. • If r-1>digits, the field is zero padded.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	r	s	4	r	+9(r-s-2).9(s)	• If r-s-2<digits are required, the field is set to all asterisks. • If r-s-2>digits, the field is zero padded.
numeric	Automatic	0	4	11	+9(10)	
numeric	Automatic	s	4	12-p+s	+9(10-p).9(s)	

Information source field with capture type packed decimal

Emitted Business Information			Information Source		COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	
numeric	r	0	n	p	r	+9(r-1)	• If r-1<digits required, the field is set to all asterisks. • If r-1>digits, the field is zero padded.
numeric	r	s	*n*	p	r	+9(r-s-2).9(s)	• If r-s-2<digits required, the field is set to all asterisks. • If r-s-2>digits, the field is zero padded.
numeric	r	Automatic	n	p	r	+9(r–p–2).9(p)	• If r-p-2<digits required, the field is set to all asterisks. • If r-p-2>digits, the field is zero padded.

Emitted Business Information			Information Source		COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	
numeric	Automatic	0	*n*	p	*n**2-p	+9(*n**2-1-p)	• If p>=*n**2-1, the field is formatted as +9(1). • When CICS is processing a capture specification with a field of this type, *n* is taken from the current capture specification. • When exporting an event specification, *n* is taken from the first capture specification.
numeric	Automatic	s	*n*	p	1+*n**2-p +s	+9(*n**2-1-p).9(s)	• If p>=*n**2-1, the field is formatted as +9(1).9(s). • When CICS is processing a capture specification with a field of this type *n* is taken from the current capture specification. • When exporting an event specification, *n* is taken from the first capture specification.

Emitted Business Information			Information Source		COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	
numeric	Automatic	Automatic	n	p	1+n *2+p	+9(n *2–1–p).9(p)	• If p>=*n**2-1, the field is formatted as +9(1).9(p). • When CICS is processing a capture specification with a field of this type *n* is taken from the current capture specification. • When exporting an event specification, *n* is taken from the first capture specification.
numeric	Automatic	0	0	p			• When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted, with precision 0. • You cannot export an event specification with a field of this type with a length of zero.
numeric	Automatic	s	0	p			• When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted, with precision s. • You cannot export an event specification with a field of this type with a length of zero.

Emitted Business Information			Information Source		COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	
numeric	Automatic	Automatic	0	p			• When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted, with precision p. • You cannot export an event specification with a field of this type with a length of zero.

Information source field with capture type zoned decimal

The CICS EP adapters include support for zoned decimal data whose sign position or representation has been specified using the COBOL SIGN clause (SEPARATE, LEADING, LEADING SEPARATE, TRAILING, and TRAILING SEPARATE).

Emitted Business Information			Information Source		COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	
numeric	r	0	*n*	p	r	+9(r-1)	• If r-1<digits required, the field is set to all asterisks. • If r-1>digits required, the field is zero padded.
numeric	r	s	*n*	p	r	+9(r-s-2).9(s)	• If r–s–2<digits required, the field is set to all asterisks. • If r–s–2>digits required, the field is zero padded.
numeric	r	Automatic	n	p	r	+9(r–p–2).9(p)	• If r–p–2<digits required, the field is set to all asterisks. • If r–p–2>digits required, the field is zero padded.

Emitted Business Information			Information Source		COBOL Copybook		
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	**Comments**
numeric	Automatic	0	*n*	p	1+*n*–p	+S9(*n*–p)	• If p>=*n*, the field is formatted as +9(1). • When CICS is processing a capture specification with a field of this type, *n* is taken from the current capture specification. • When exporting an event specification, *n* is taken from the first capture specification.
numeric	Automatic	s	*n*	p	1+*n*–p+s	+9(*n*–p).9(s)	• If p>=*n*, the field is formatted as +9(1).9(s). • When CICS is processing a capture specification with a field of this type, *n* is taken from the current capture specification. • When exporting an event specification, *n* is taken from the first capture specification.

Emitted Business Information			Information Source		COBOL Copybook		
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	**Comments**
numeric	Automatic	Automatic	*n*	p	1+*n* +p	+9(*n* –p).9(p)	• If p>=*n*, the field is formatted as +9(1).9(p). • When CICS is processing a capture specification with a field of this type, *n* is taken from the current capture specification. • When exporting an event specification, *n* is taken from the first capture specification.
numeric	Automatic	0	0	p			• When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted, with precision 0. • You cannot export an event specification with a field of this type with a length of zero.
numeric	Automatic	s	0	p			• When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted, with precision s. • You cannot export an event specification with a field of this type with a length of zero.

Emitted Business Information			Information Source		COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Precision**	**Length**	**Format**	
numeric	Automatic	Automatic	0	p			• When CICS is processing a capture specification with a field of this type, all of the data that is available is emitted, with precision p. • You cannot export an event specification with a field of this type with a length of zero.

Information source field with capture type short float

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
scientific	r	0	4	r	X(r)	• If r<7, the field is set to all asterisks. • Data format is +9E +999 and is padded on the right with blanks to length r.
scientific	r	s	4	r	X(r)	• If r<7, the field is set to all asterisks. • If r<s+8, the precision is reduced as necessary. • Data format is +9.9(s)E+999 and is padded on the right with blanks to length r.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
scientific	r	Automatic	4	r	X(r)	• If r<7, the field is set to all asterisks. • If r<14, the precision is reduced as necessary. • Data format is +9.9(6)E+999 and is padded on the right with blanks to length r.
scientific	Automatic	0	4	7	X(7)	Data format is +9.9E +999.
scientific	Automatic	s	4	s+8	X(s+8)	Data format is +9.9(s)E+999.
scientific	Automatic	Automatic	4	14	X(14)	Data format is +9.9(6)E+999.
numeric	r	0	4	r	+9(r–1)	• If r–1<digits required, the field is set to all asterisks. • If r–1>digits required, the field is zero padded.
numeric	s	r	4	r	+9(r–s–2).9(s)	• If r–s–2<digits required, the field is set to all asterisks. • If r–s–2>digits required, the field is zero padded.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	r	Automatic	4	r	X(r)	When CICS is processing a capture specification with a field of this type, the number is formatted with as much precision as possible within the format length r. If the number of whole number digits that is required exceeds r, the field is set to all asterisks.
numeric	Automatic	0	4			When CICS is processing a capture specification with a field of this type, the number is formatted with as much precision as possible within the format length r. You cannot export an event specification with a field of this type.
numeric	Automatic	s	4			• When CICS is processing a capture specification with a field of this type, the number is formatted with: – A sign – As many whole digits as required – s digits after the decimal point. • You cannot export an event specification with a field of this type.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	Automatic	Automatic	4			• When CICS is processing a capture specification with a field of this type, the number is formatted with: – A sign – As many whole digits as required – As many digits after the decimal point as required. • You cannot export an event specification with a field of this type.

Information source field with capture type long float

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
scientific	r	0	8	r	X(r)	• If r<7, the field is set to all asterisks. • Data format is +9E +999 and is padded on the right with blanks to length r.
scientific	r	s	8	r	X(r)	• If r<7, the field is set to all asterisks. • Data format is +9.9(s)E+999 and is padded on the right with blanks to length r. • If r<s+8, the precision is reduced as necessary.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
scientific	r	Automatic	8	r	X(r)	• If r<7, the field is set to all asterisks. • If r<23, the precision is reduced as necessary. • Data format is +9.9(15)E+999 and is padded on the right with blanks to length r.
scientific	Automatic	0	8	7	X(7)	Data format is +9.9E +999.
scientific	Automatic	s	8	s+8	X(s+8)	Data format is +9.9(s)E+999.
scientific	Automatic	Automatic	8	23	X(23)	Data format is +9.9(15)E+999.
numeric	r	0	8	r	+9(r–1)	• If r–1<digits required, the field is set to all asterisks. • If r–1>digits required, the field is zero padded.
numeric	s	r	8	r	+9(r–s–2).9(s)	• If r–s–2<digits required, the field is set to all asterisks. • If r–s–2>digits required, the field is zero padded.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	r	Automatic	8	r	X(r)	When CICS is processing a capture specification with a field of this type, the number is formatted with as much precision as possible within the format length r. If the number of whole number digits that is required exceeds r, the field is set to all asterisks.
numeric	Automatic	0	8			When CICS is processing a capture specification with a field of this type, the number is formatted with as much precision as possible within the format length r. You cannot export an event specification with a field of this type.
numeric	Automatic	s	8			• When CICS is processing a capture specification with a field of this type, the number is formatted with: – A sign – As many whole digits as required – s digits after the decimal point. • You cannot export an event specification with a field of this type.

Emitted Business Information			Information Source	COBOL Copybook		Comments
Type	**Format length**	**Format precision**	**Length**	**Length**	**Format**	
numeric	Automatic	Automatic	8			• When CICS is processing a capture specification with a field of this type, the number is formatted with: – A sign – As many whole digits as required – As many digits after the decimal point as required. • You cannot export an event specification with a field of this type.

CICS container-based event (CCE) format

The CICS container-based event (CCE) format is based on a CICS channel with containers providing the event name and details, standard contextual information, and the event-specific data items. The CCE format can be used to extend application behavior in an event-driven way by driving a new CICS transaction, which uses the information passed to it in the container-based event format.

You specify the event format you want to use by using the event binding editor. For more information about adapter properties and supported formats, see Specifying EP adapter and dispatcher information in the CICS Explorer product documentation.

The CCE format is used to pass the event data to a CICS program. The formatted data is placed in containers in a channel named `DFHEP.EVENT`.

Some context information for correlation is placed in the container named `DFHEP.CCECONTEXT`. For more information, see Table 20 on page 97.

The name of each data item is placed in containers named `DFHEP.NAME.`*nnnnn*, where *nnnnn* is an ascending number starting with 00001.

The formatted data is in containers named `DFHEP.DATA.`*nnnnn*, where *nnnnn* is an ascending number starting with 00001. This number correlates the name of the data item to the formatted data.

The `DFHEP.CCECONTEXT` container contains the formatted event context information, which can be mapped by using the `EPFEContextEntry` structure.

Table 20. EPFEContextEntry structures

Name	Language	Library
DFHEPFED	Assembler	SDFHMAC
DFHEPFEH	C/C++	SDFHC370

Table 20. EPFEContextEntry structures (continued)		
Name	**Language**	**Library**
DFHEPFEL	PL/I	SDFHPL1
DFHEPFEO	COBOL	SDFHCOB

The copybooks providing these mappings can be found here:

- hlq.SDFHMAC
- hlq.SDFHC370
- hlq.SDFHPL1
- hlq.SDFHCOB

The EPFE structure consists of contextual data associated with the event which includes:

- The event context identifier.
- The event context version.
- The event context event binding name.
- The event binding user tag.
- The event business name.
- The network unit of work ID[1].
- The network qualified applid.
- The capture date and time[2].
- The capture specification name.
- The item count[3].

Note:

[1] The network unit of work ID is set to binary zeros (NULL) for all system and policy events.

[2] All events are emitted with Coordinated Universal Time (UTC) timestamps.

[3] The number of DFHEP.NAME.nnnnn and DFHEP.DATA.nnnnn containers.

Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 6 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- CICS TS security
- Developing for external interfaces
- Application development reference
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 6 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- CICS TS diagnostics reference

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 6 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services
- Customization Guide

- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 6 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the U.S. and other countries.

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

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights
Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

For CICS Explorer:
Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user's user name and password, for purposes of session management,

authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled *Cookies, Web Beacons and Other Technologies* and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index