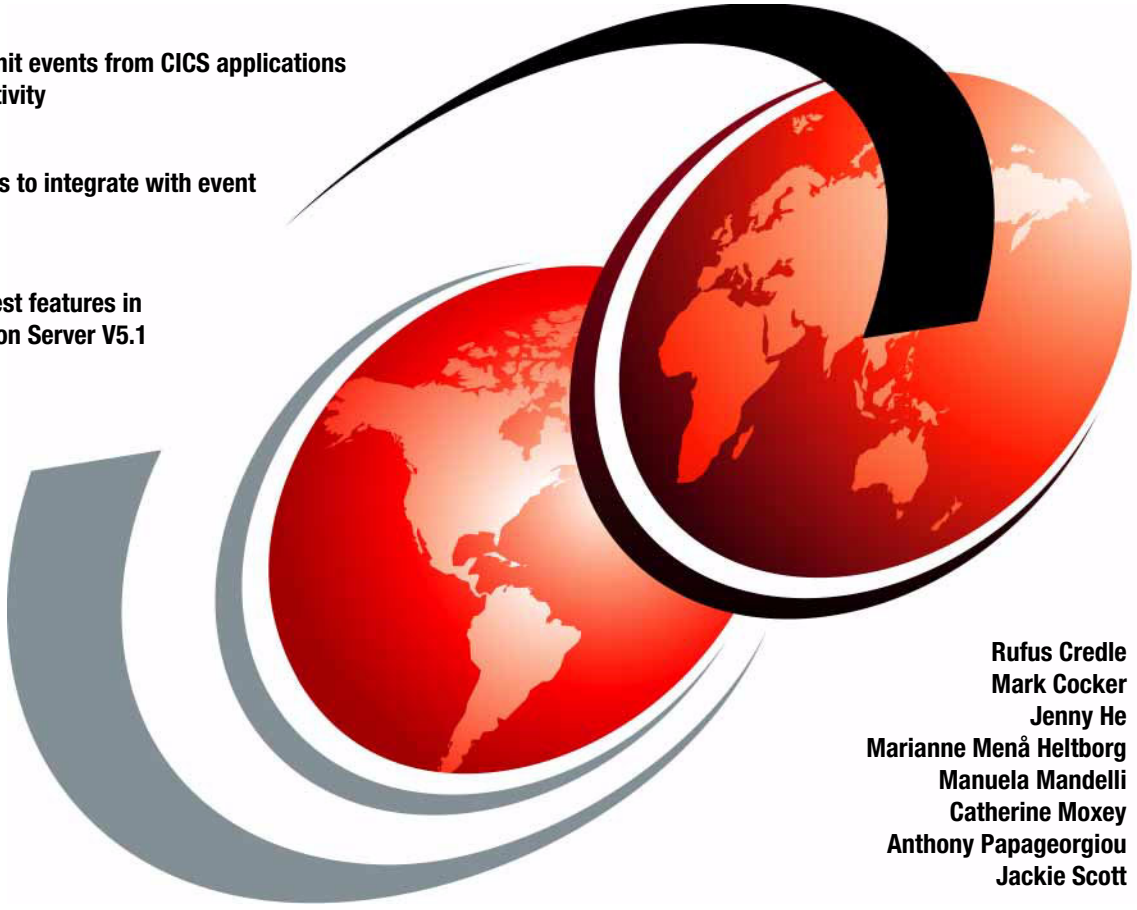
IBM

Event Processing with CICS

- Capture and emit events from CICS applications and system activity
- Use CICS events to integrate with event consumers
- Review the latest features in CICS Transaction Server V5.1



Rufus Credle
Mark Cocker
Jenny He
Marianne Menå Heltborg
Manuela Mandelli
Catherine Moxey
Anthony Papageorgiou
Jackie Scott

Redbooks

ibm.com/redbooks

International Technical Support Organization

Event Processing with CICS

August 2013

SG24-7792-01

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

Second Edition (August 2013)

This edition applies to Version 5, Release 1, CICS Transaction Server.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

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 right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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 the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

BladeCenter®
CICS Explorer®
CICSPlex®
CICS®
Cognos®
DB2®
IBM®
ILOG®
MVS™
OMEGAMON®
OS/390®
RACF®
Rational®
Redbooks®
Redguide™
Redpapers™
Redbooks (logo) ®
System x®
System z®
Tivoli®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

Evolution, and Kenexa device are trademarks or registered trademarks of Kenexa, an IBM Company.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Preface

This completely refreshed IBM Redbooks® publication provides a detailed introduction to the latest capabilities for business event processing with IBM® CICS® V5. Events make it possible to identify and react to situations as they occur, and an event-driven approach, where changes are detected as they happen, can enable an application or an Enterprise to respond in a much more timely fashion. CICS event processing support was first introduced in CICS TS V4.1, and this IBM Redbooks® publication now covers all the significant enhancements and extensions which have been made since then.

CICS Transaction Server for z/OS provides capabilities for capturing application events, which can give insight into the business activities carried out within CICS applications, and system events, which give insight into changes in state within the CICS system. Application events can be generated from existing applications, without requiring any application changes.

Simple tooling allows both application and system events to be defined and deployed into CICS without disruption to the system, and the resulting events can be made available to a variety of event consumers. CICS events can amongst other things be used to drive processing within CICS, to populate dashboards that are provided by IBM Business Monitor and to search for patterns in events using IBM Operational Decision Manager.

This IBM Redbooks® publication is divided into the following parts:

Part 1 introduces event processing. We explain what it is and why you need it, and discuss how CICS makes it easy to both capture and emit events.

Part 2 of the book focuses on the details of event processing with CICS. It gives a step-by-step guide to implementing CICS events, along with the environment used in the examples.

Part 3 provides some guidance on governance and troubleshooting for CICS events, and describes how to integrate CICS events with IBM Operational Decision Manager and IBM Business Monitor.

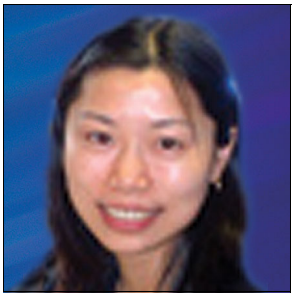
The Appendices include additional reference information.

Authors

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

Rufus Credle is a Certified Consulting IT Specialist at the ITSO, Raleigh Center. In his role as project leader, he conducts residencies and develops IBM Redbooks and Redpapers™. Subjects include network operating systems, enterprise resource planning (ERP) solutions, voice technology, high availability, clustering solutions, web application servers, pervasive computing, IBM and OEM e-business applications, WebSphere Commerce, IBM industry technology, System x®, and IBM BladeCenter®. Rufus' various positions during his IBM career include assignments in administration and asset management, systems engineering, sales and marketing, and IT services. He has a BS degree in Business Management from Saint Augustine's College. Rufus has been employed at IBM for 33 years.

Mark Cocker is a senior software engineer in the CICS Strategy and Planing team in IBM, based at Hursley Laboratory, England. Mark has 20 years of experience in CICS development, service, beta programs and the IBM Design center. He holds a degree in Information Systems Management from Bournemouth University and is an IBM Certified SOA Associate and Solution Designer - CICS Enablement for e-business. His areas of expertise include enterprise connectivity, event processing, and Java. He has written several CICS SupportPacs, papers, and presents CICS topics regularly at conferences.

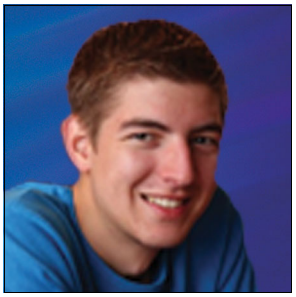


Jenny He is a Software Engineer in the CICS Transaction Server product suite at Hursley, UK. Jenny has worked at IBM for 10 years and has broad experience in IBM software development, agile development process, solution building, and testing. Jenny holds a Bachelor of Engineering degree in Electronics from South China University of Technology in China, and a PhD degree in optical networking from the University of Essex in the UK. Her areas of expertise on CICS include event processing and policy management.

Marianne Menå Heltborg is a Consulting IT Specialist, IBM Certified working in IBM Software Group, Denmark. She joined IBM in 1986 and originally worked as a CICS Systems Programmer. She has been involved in implementing SOA for several major clients, particularly in the areas of Enterprise Modernization and Application Integration. Her areas of expertise include the CICS TS and the WebSphere product suite running on IBM z/OS®.

Manuela Mandelli is a Senior Product Specialist in IBM Italy, based in Vimercate, Milan. She has 25 years of experience in supporting CICS. She studied in Liceo Scientifico in Milano. Her areas of expertise include CICS Transaction Server, CICS Transaction Gateway and IBM Session Manager.

Catherine Moxey is an IBM Senior Technical Staff Member in the CICS Transaction Server for IBM z/OS team, based in Hursley, UK. She holds an M.A. in Chemistry from Oxford University, and has has nearly 30 years of experience as a software engineer, 24 of those with IBM. Her areas of expertise include CICS, IBM System z®, Event Processing and Cloud enablement, and she is the architect for event processing support in CICS. Catherine is a member of the Event Processing Technical Society where she is active in the Reference Architecture workgroup and has presented at a number of industry conferences on Event Processing.



Anthony Papageorgiou is a Software Developer in the CICS Transaction Server development organization based at the Hursley laboratory in the UK. Before he joined IBM in 2007, he worked for Mintel International Group Ltd. as a Web Developer. Anthony has a Bachelor of Science degree in Computer Science from the University of Warwick. His areas of expertise include Web 2.0, Event Processing and Mobile Technologies, and he was closely involved with the design and development of their support in CICS V4.

Jackie Scott is a Software Test Engineer at the Hursley laboratory, England and has more than 25 years of experience in the IT industry, starting out as a programmer in a mainframe environment. Jackie joined IBM as a Software Support Specialist providing customer support for IBM MVS™ and IBM OS/390®, followed by several years as an MVS systems programmer. After four years in the Java Technology Centre at Hursley, Jackie joined CICS Development (also at Hursley) as a Software Tester and has worked primarily on Event Processing.

Thanks to the following authors of the previous editions of this book:

- Chris Rayns
- Michael Baylis
- Ann Colins
- Hannah Said
- Phil Bareham
- Steve Bolton
- Lee Gavin
- Phil Lee
- Jackie Scott
- Tommy Joergensen

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

- Tamikia Barrow-Lee
- Richard Conway
- Robert Haimowitz

  International Technical Support Organization, Raleigh and Poughkeepsie Center

- Chris Backhouse, IBM ILOG® Synergies and Integration Team

  IBM Hursley

- Daniel Millwood, Software Developer

  IBM Hursley

- Mark Hiscock, WebSphere Business Events Development

  IBM Hursley

- Peter Crocker, Development Lead, Architect, Business Event Processing

  IBM Hursley

Francis Burgess, Software Product Introduction Specialist
IBM Hursley

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions while getting hands-on experience with leading-edge technologies. You have the opportunity to team with IBM technical professionals, Business Partners, and clients.

Your efforts help increase product acceptance and customer satisfaction. As a bonus, you develop a network of contacts in IBM development labs and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at this website:

http://ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online Contact us review Redbooks form found at this website:

  http://ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Part 1

Introduction

In this part of this IBM Redbooks publication, we introduce event processing, explain what it is, and why you need it. We also review the CICS implementation of event processing.

1

Introduction to event processing and CICS event processing support

This chapter describes the concept of event processing and explains the various elements of an event processing solution. It explains why event processing is useful to your business, and includes a summary of IBM products for event processing solutions.

Also introduced in this chapter are the capabilities of CICS Events, and a description of the evolution of those capabilities over a number of CICS releases.

This chapter includes the following topics:

- What is an event?
- What is event processing?
- Why you need events
- Business application events and system events
- IBM solutions for business event processing
- Introduction to CICS event processing support
- Evolution of CICS event processing support
- Summary

1.1 What is an event?

An event is something that happens that is significant to a system. Events include the following examples:

- Open a bank account.
- Sense a temperature change.
- Click a mouse button.
- Detect loss of connectivity to a database.
- Browse an inventory without making a purchase.
- An unusual history of purchases on a credit card.
- An increasing frequency of application failures.

For this book, *event* is the term that is used to describe an electronic message that indicates a change in the state of some aspect of a system. An event has a name and usually some data, which sometimes referred to as the *event payload*.

Events are generated and processed independently and in near real time. The processing of an event is de-coupled from the computer operations that cause it to be emitted.

1.2 What is event processing?

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 usage of the processed events.

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 then complete or fail. The activity of the enterprise and its business can be monitored and changed as a result of events. Event processing consists of the following main tasks:

- Event sources emit events into the event processing system. Examples of event sources are simple Radio Frequency Identification (RFID) sensors and actuators, business flows, and CICS applications.
- Some processing is carried out on one or more events. An event processing system can perform the following various actions on events:
  - Simple enriching of the event (for example, adding a time stamp to the event data).
  - Adding information about the source of the event, such as the context in which the event occurred.

  - Processing multiple simple events, from multiple event sources, against event patterns to produce a new derived event. Processing of this kind is often referred to as *complex event processing*.

  The event that results from event processing is made available for consumption.

- Event consumers react to events. The event consumer might be simple and only update a database or a visual dashboard with the data that is carried with the event, or it might carry out new business processing that is based on the event.

An event source or event consumer might carry out some of the simple types of event processing, while more complex event processing (such as looking for patterns across multiple events) is carried out by some separate event processing component.

Figure 1-1 on page 6 is a simplified version of the conceptual model for Event Processing, which is described in the IBM Redguide™ *A Conceptual Model for Event Processing Systems*, REDP-4642-00, which is available at this website:

`http://www.redbooks.ibm.com/abstracts/redp4642.html?Open`

The conceptual model defines the various components that can be involved in an event processing solution. However, the only required components are some kind of Event Emitter to emit events that are generated by event producers, an Event Handler layer to handle events for consumption by event consumers, and an Event Bus to deliver events from producers to consumers and potentially carry out event processing and other services with the events.

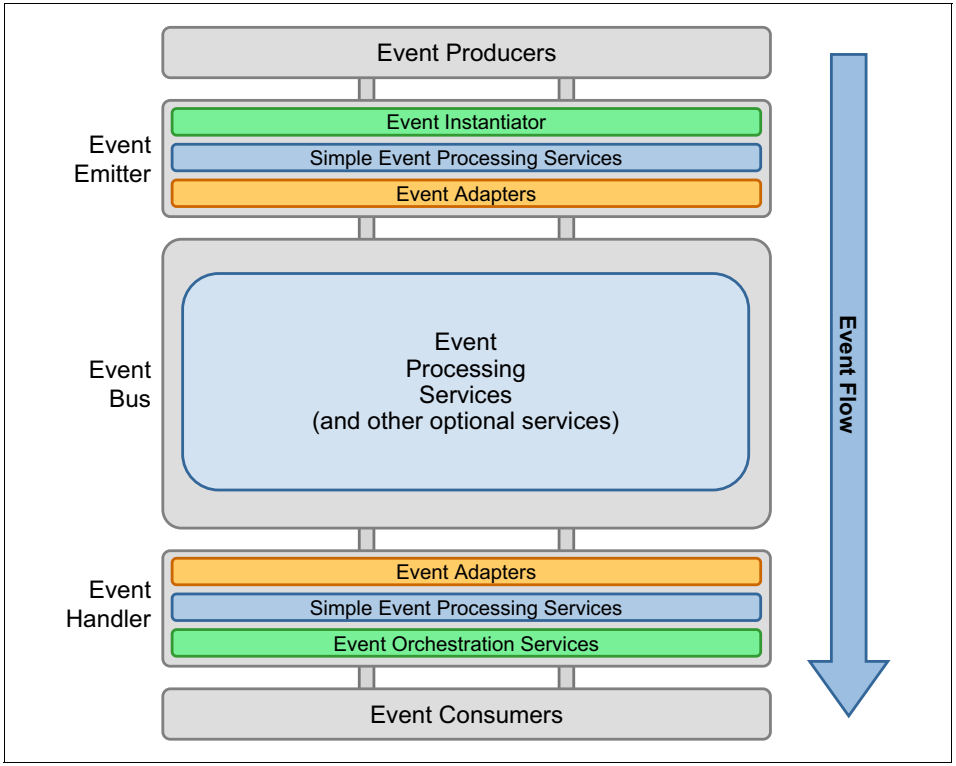


Figure 1-1 Event Processing conceptual model

CICS can act as an Event Emitter and can carry out simple event processing such as filtering, capture, enrichment, formatting, and routing of those single business events. The event producers in this case are CICS applications or the CICS system.

The following section introduces the concepts of simple and complex event processing, and events are sometimes categorized into simple and complex events.

1.2.1 Simple events

The first four examples in the list in section 1.1, “What is an event?” on page 4 are called *simple events*. For many years, organizations used simple event processing to detect and respond to discrete events (for example, when customers open a new account sending them a welcome letter that explains other facilities that are provided by the bank).

1.2.2 Complex events

The last three examples in the list in section 1.1, “What is an event?” on page 4 describe what are often called *complex* or *derived* events, which are obtained by looking at the patterns of simple events over time. To detect customers browsing an inventory without making a purchase, a system could emit events when customers perform the following actions:

- Browse an item in the inventory.
- Buy an item.

Detecting the patterns from these simple events can produce the complex event *Browse an inventory without making a purchase,* as shown in section 1.1, “What is an event?” on page 4.

In large organizations, countless events occur every day, but not all events are of equal importance. Greater insight can be obtained when a pattern of related or seemingly unrelated events from one or more sources is detected, and responses to that pattern are coordinated. Considerable complexity, time, and cost are involved in writing custom code for such a solution. This can be replaced by general software technology that is designed to detect event patterns and coordinate responses, sometimes known as *complex event processing software*.

1.3 Why you need events

Events make it possible to identify and react to situations as they occur. An event-driven approach, where changes are detected as they happen, enable an application or an Enterprise to respond in a much more timely fashion. Events can give insight into what is happening within an application or within a computing system. Event processing also can provide a way of extending an existing application in a flexible and noninvasive manner.

By using events to drive certain actions or more processing, it is easy to change those actions or to change the situations in which they are carried out without affecting the underlying application processing that produces the events.

An aspect of event processing that is gaining considerable momentum is a focus on the business value that can be obtained from events based on the growing need to react and make decisions much closer to real time, and to gain insight into business processing. The drivers behind this momentum include the introduction of compliance regulations that require real-time responses to situations, and the desire to respond rapidly to changes in the business without entailing long development cycles.

An important aspect of Events is the 'Principle of Decoupling': the producer of an event (the event source) and the consumer (or consumers) of an event are decoupled from each other. The event producer does not rely on the event consumer that is carrying out any particular processing (or any processing at all) and does not need to wait for a response from the consumer. Similarly, the event consumer does not depend on any processing that is carried out by the event producer, apart from producing the actual event. This decoupling enables a more flexible and agile approach to building and extending applications.

1.4 Business application events and system events

All events in an enterprise can be seen as having a business consequence and so can be described as business events. Whether the event and event processing are specified by a line-of-business manager or an IT programmer, the event relates to the business application or system that is running. Therefore, it has business relevance. For example, an event that implies imminent failure of a system that is running an order-processing application could be considered relevant to the line-of-business manager in the same way that an event about the processing of an order is relevant. This book regards all events as having the potential to be business events, but makes a distinction between system and application events.

System events generally have a technical focus and relate to monitoring the operating system, running applications, and middleware that is running on the system. The event data is usually equally technical, specifying the identifiers of the resources under observation. In CICS terms, system events are captured from processing that is carried out by the CICS system.

Application events are usually related to higher-level business processes. They specify conditions in terms of what the application is doing for the business. For example, contrast the business event “a new order has been placed” with the system events that are used to assess the compute time for processing the order: inventory file opened, order information storage released, and so forth. In CICS terms, application events are captured from processing that is carried out by business applications.

The consumers of business events are often required to be independent of the implementation specifics of the systems that emit the events. For example, it is possible for one event consumer to process events from several disparate ordering systems and provide a single consolidated view of the business application’s state.

CICS event processing produces application and system events, but the audience and uses for these two classes of events might differ.

1.5 IBM solutions for business event processing

The IBM software portfolio enables a range of options for integrated processing of business events. The following options are described in this section:

- CICS Transaction Server
- CICS Explorer
- TXSeries Events SDK
- IBM Operational Decision Manager
- IBM Business Monitor
- Cognos Real-time Monitoring
- WebSphere Message Broker
- WebSphere Enterprise Service Bus
- Tivoli OMEGAMON XE for CICS on z/OS
- Solutions reviewed

1.5.1 CICS Transaction Server

IBM CICS Transaction Server (TS) business applications are the main source of business information in most large enterprises. The CICS run time detects instances of events that are enabled and captures the events and payload without the need to make application code changes or to provide system code.

CICS event processing is a core component of the CICS run time and provides all the qualities of service you would expect of CICS. It is possible to emit events in formats that are suitable for use by IBM Operational Decision Manager Events, IBM Business Monitor, and other users.

CICS Event Processing support is extensible, with options for customization.

For more information, see *CICS Transaction Server for z/OS, Version 5 Release 1* at this website:

`http://publib.boulder.ibm.com/infocenter/cicsts/v5r1/index.jsp`

1.5.2 CICS Explorer

IBM CICS Explorer® provides a modern, powerful interface for managing CICS systems and resources and is an integration point for CICS and other tooling. The CICS Explorer provides a common, intuitive, and Eclipse-based environment for architects, developers, administrators, system programmers, and operators.

CICS Explorer includes the following features:

- Task-oriented views that provide integrated access to a broad range of data and control capabilities.
- Powerful, context-sensitive resource editors.
- An integration point for CICS TS, CICS Tools, CICS Transaction Gateway, Problem Determination Tools, and IBM Rational® Tools. It is included in Rational Developer for IBM System z.

For CICS event processing, the CICS Explorer provides the CICS event binding editors, which are used to perform the following tasks:

- Define events to be emitted and their payload data.
- Specify to the CICS run time how to detect when the events occur.
- Indicate how events are to be formatted and routed.
- Deploy the event definitions to zFS for installation into CICS, by using CICS Bundle support.

CICS Explorer V5.1 includes full support for the event processing capabilities that are described in this Redbooks publication. For more information, see this website:

http://www.ibm.com/software/htp/cics/explorer/

1.5.3 TXSeries Events SDK

IBM TXSeries for Multiplatforms is a transaction-processing monitor that runs on distributed platforms and supports CICS APIs. SupportPac CB01 provides a software development kit for emitting events from applications that are running in TXSeries. It emits application events in the event format that is used by the events component of IBM Operational Decision Manager and supports WebSphere MQ or HTTP as the event transport. Unlike CICS TS application events, TXSeries event support does require a small application change, but is easy to use and handles all of the event formatting and routing for you.

This IBM Redbooks publication does not describe the TXSeries support for events further.

SupportPac CB01 can be downloaded from the following website:

http://www.ibm.com/support/docview.wss?uid=swg24025837

1.5.4 IBM Operational Decision Manager

IBM Operational Decision Manager (ODM) integrates business events with business rules to enable decision making in real time. IBM ODM provides the power to intelligently automate a wide range of decisions across business processes and applications, which unites market-leading IBM capabilities for business rules management and business event processing to enable more responsive actions to business opportunities or risk conditions.

The event processing component of ODM supports business event processing by meeting the high-volume demands and processing that is required across industries and application domains. ODM provides graphical, codeless user interfaces that simplify implementation and empower business users to develop and maintain event pattern detection logic.

Based on user definitions, the ODM event server engine detects and sifts though the mass of events that occur across the information infrastructure and identifies only those events and patterns of interest.

For more information, see this website:

http://www.ibm.com/software/decision-management/operational-decision-management/

1.5.5 IBM Business Monitor

IBM Business Monitor is a comprehensive business activity monitoring (BAM) product that provides business users and managers with a real-time and end-to-end view of business processes, events, and operations. IBM Business Monitor aggregates and correlates events into metrics that give objective measurements on the status of business processes.

IBM Business Monitor shows business users real-time information about the performance of critical business processes. It offers customizable dashboards that enable complete insight into the business flowing through the system. These dashboards can calculate and display key performance indicators (KPIs) and metrics derived from the following sources:

- Business processes
- Business activity data
- Business events

Business users can view these KPIs, metrics, events, and alerts through various means, including lightweight web interfaces, smartphones, corporate portals, and on desktops. These options give business users immediate actionable information and insight into their business operations to mitigate risk and take advantage of opportunities.

For more information, see this website:

http://www.ibm.com/software/integration/business-monitor/

1.5.6 Cognos Real-time Monitoring

IBM Cognos® Real-time Monitoring is an operational Business Intelligence solution that provides easy access to consistent data, which allows you to react quickly to revenue and cost-saving opportunities. Cognos Real-time Monitoring features self-service, interactive dashboards with operational key performance indicator measures for frontline business users, executives, managers, and analysts.

For more information about Cognos Real-time Monitoring and IBM Business Intelligence solutions, see this website:

http://www.ibm.com/software/products/us/en/cognos-real-time-monitoring/

1.5.7 WebSphere Message Broker

IBM WebSphere Message Broker is a lightweight, advanced Enterprise Service Bus (ESB) that enables the integration of data sources from various platforms across service-oriented architecture (SOA) and non-SOA environments.

IBM WebSphere Message Broker is built for universal connectivity and transformation in heterogeneous IT environments, and makes use of the power and reliability of IBM WebSphere MQ. It distributes information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond.

With WebSphere Message Broker, organizations of any size can eliminate point-to-point connections and batch processing to increase business flexibility and smart interoperability of systems regardless of platform, protocol, or data format.

For more information, see this website:

http://www.ibm.com/software/integration/wbimessagebroker/

1.5.8 WebSphere Enterprise Service Bus

IBM WebSphere Enterprise Service Bus provides integration for new and composite service-oriented architecture (SOA) and Business Process Management (BPM) applications, optimized for the WebSphere Application Server platform. It decouples complex integration logic from applications and can provide consistency in connectivity across a diverse infrastructure.

WebSphere Enterprise Service Bus can provide service mediation and hosting on common internet-standard application infrastructures that are based on WebSphere Application Server.

WebSphere Enterprise Service Bus provides a smart approach to SOA, which delivers a standards-based connectivity and integration solution that allows you to create and deploy interactions quickly and easily between applications and services, with a reduced number and complexity of interfaces.

For more information, see this website:

http://www.ibm.com/software/integration/wsesb/

1.5.9 Tivoli OMEGAMON XE for CICS on z/OS

IBM Tivoli® OMEGAMON® XE for CICS on z/OS enables monitoring and management of CICS transactions and resources. It quickly detects and isolates problems when they occur on your complex CICS systems to minimize or eliminate any effect on your customers and business.

Tivoli OMEGAMON XE for CICS provides events for a broader and more advanced range of state changes and system situations than the CICS system events that are described in this Redbooks publication. However, it uses a polling mechanism to discover state changes, which implies a bigger effect on CICS system performance and a less timely reporting of events. CICS system events are a targeted set of event conditions that particularly benefit from not using polling.

Tivoli OMEGAMON also enables integration with system events from other systems.

For more information about IBM Tivoli OMEGAMON XE for CICS, see this website:

http://www.ibm.com/software/tivoli/products/omegamon-xe-cics/

1.5.10 CICS SupportPac CA1Y: Send email from CICS TS for z/OS

SupportPac CA1Y provides an SMTP client to emit events that are captured in CICS as emails. Use the CICS Explorer event adapter or event binding editors to define the SupportPac transaction CA1Y as the custom event processing adapters (EP adapters), and specify the email headers, content, attachments, and server details. Information from the event can be merged in with predefined content templates for added flexibility.

The SupportPac runs in the CICS JavaServer environment and is eligible for offloading onto a System z Application Assist Processor (zAAP) speciality engine, which might be cost-effective compared to alternatives.

For more information, see this website:

http://www.ibm.com/support/docview.wss?uid=swg24033197

1.5.11 Solutions reviewed

This section provides a review of the products and the situations in which they should be used.

If your business processing is running within CICS, that is the source of your events and forms the subject of this IBM Redbooks publication. There are situations in which the actions to be taken as a result of the events also involve processing within CICS. In other situations, you want to involve other products.

If you want to monitor the processing that is happening within CICS, look at key performance indicators, provide a dashboard to allow business users to understand the behavior of the business, and receive alerts, you could use IBM Business Monitor or IBM Cognos Real-time Monitoring.

If you want to derive more information from combinations of events, potentially including events from other sources in addition to CICS, or consider events over time, you can use IBM Operational Decision Manager.

You might also want to monitor processing based on some of these derived events, in which case IBM Operational Decision Manager can look for the patterns of interest and can send the resulting events to IBM Business Monitor.

If you want to transform and enrich events from CICS and make them available to a range of consumers, you could use WebSphere Message Broker or WebSphere Enterprise Service Bus to provide transformation and mediation.

For more information, see *Leveraging CICS Events with an ESB*, SG24-7863-00, which is available at this website:

http://www.redbooks.ibm.com/abstracts/sg247863.html

1.6 Introduction to CICS event processing support

Figure 1-2 provides an overview of CICS event processing support.

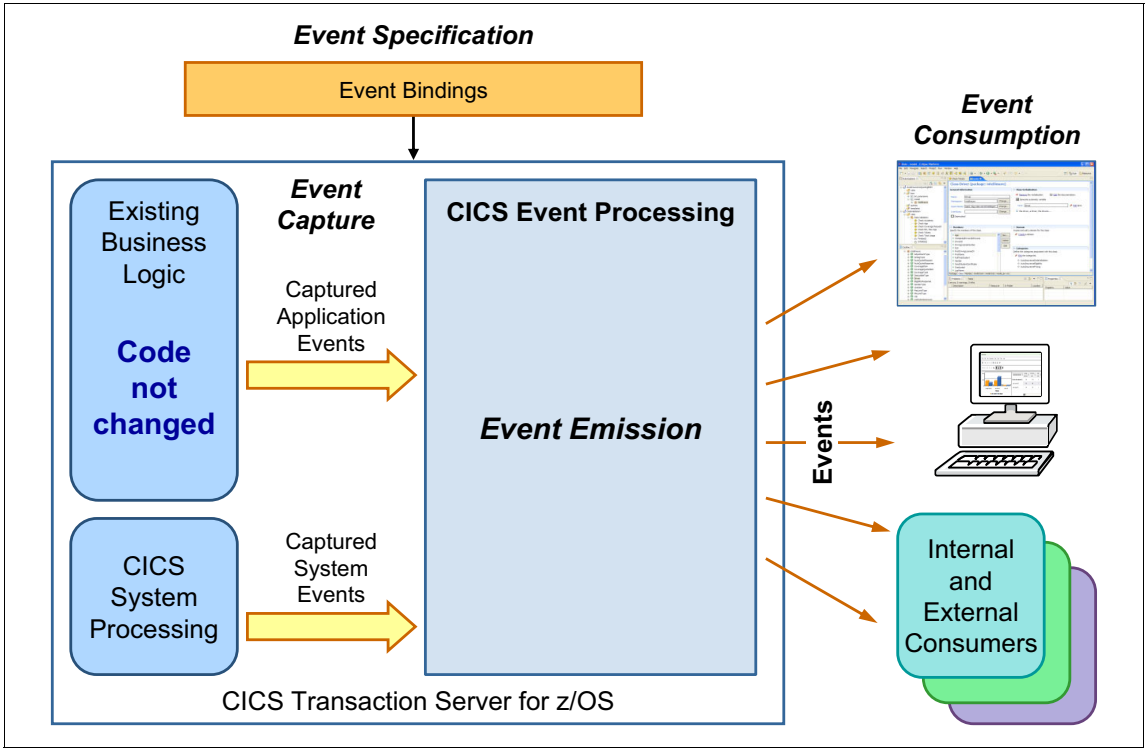


Figure 1-2 Overview of CICS event processing support

CICS Transaction Server for z/OS can act as a source of application and system events. In both cases, the events that are to be produced are specified to CICS by installing and enabling event bindings by using development and deployment tools. An event binding specifies the situations in which the events are to be captured, the data to be included in the events, and how the events are to be emitted. An event binding is an XML document that can be created and edited by using the Event Binding Editor, a component of the CICS Explorer.

Events can be captured from existing applications without requiring any changes to the application, or from CICS system processing without the need to poll for changes in stage. CICS carries out some simple event processing on the captured event, such as filtering and enrichment, and the event can then be formatted and routed (emitted) to a range of event consumers. CICS event processing provides all the expected CICS qualities of service, including security, performance, diagnostics and monitoring. Events from CICS can be used by many different kinds of event consumers, ranging from transactions that are running within CICS to external event monitors and decision servers.

A capture specification within an event binding provides the information about the situation in which an event is to be captured. An EP adapter that is referenced by an event binding (or embedded within an event binding) provides the information about how the events are to be emitted.

A capture specification includes a capture point (the point in CICS processing where the event can be detected), some filtering information (to identify the particular circumstances in which the event occurs), and information sources (to specify how the information to be included in the event can be obtained from data available at the capture point). The sets of capture points which can be specified depend on whether this is an application or a system event.

1.6.1 CICS application events

Application events represent interesting business situations that arise during processing that is carried out by CICS applications. Examples of application events include an order being received, a deposit being made into a bank account, or a stock trade being carried out.

Application events are *noninvasive,* meaning that events can be captured from a CICS application without the need to change or recompile the application, and with minimal if any overhead in the performance of the application.

The capture points for application events are a subset of the CICS API commands. They also are a capture point when a program is initiated and the SIGNAL EVENT command which can be coded in an application as an explicit, and in this case invasive, capture point. The CICS API commands that can be specified as capture points represent the points in application processing where business events are most likely to occur, such as when data is input or output by the application. For more information about the application event capture points, see Chapter 2, “Capturing application and system events” on page 23.

1.6.2 CICS system events

System events represent significant system changes that arise during the running of a CICS system. Examples of system events include an IBM DB2® system changing from being connected to the CICS system to not being connected, the number of tasks in a transaction class increasing to more than 90% of the limit for that class, or an unhandled application abend occurring.

CICS system events are also non-invasive in that they do not require you to write any code and the system overhead is minimal. System events are detected by the CICS system as the system changes are processed and are therefore more efficient than other technologies that rely on polling.

The capture points for system events are a subset of some of the state changes that can occur in CICS. In addition to the set of predefined state changes, the Message capture point allows events to be captured when most of the CICS messages are issued, which enables system events to be captured for any of the state changes that result in such messages. Later chapters in this book provide more information about system event capture points.

1.6.3 CICS event emission

The way in which an event is to be emitted is specified by an EP adapter. The EP adapter specifies the transport over which the event is emitted (such as WebSphere MQ, HTTP, or transaction start of a CICS transaction), the format in which the event should be emitted (such as a predefined XML format or a structured data format), and some advanced emission options (such as whether the event is transactional, or should it be emitted synchronously to the task in which it was captured). Several of the advanced options apply only to application events.

Later chapters in this book provide more information about EP adapters, including how you can write your own custom EP adapter to support any transport or formatting.

1.7 Evolution of CICS event processing support

This IBM Redbooks publication assumes the level of event functionality that is available in CICS Transaction Server for z/OS V5.1. CICS Event support was first introduced in CICS Transaction Server for z/OS (CICS TS) V4.1 and was further enhanced in versions CICS TS 4.2 and CICS TS V5.1. This section explains the levels of functionality that were introduced in each of those releases.

If you are using CICS TS V4.2 or V4.1, not all of the event functions that are described in this Redbooks publication are available to you. You are advised to use the Event Binding Editor that is included in CICS Explorer V5.1. This is compatible with previous levels of event processing support in CICS, but note that if you select options in an event binding that are only available in later releases, the event binding fails to install into an earlier CICS run time.

1.7.1 CICS event capabilities in CICS TS V4.1

The following broad set of capabilities formed the initial CICS event support in CICS TS V4.1:

- Application events captured non-invasively from a subset of the CICS API and on program initiation.
- SIGNAL EVENT application programming command.
- Event bindings, containing event specifications, event capture specifications, and EP adapters.
- Event Binding Editor in the CICS Explorer for creating and manipulating event bindings.
- EP adapters supplied by CICS for emitting events to CICS TS queues, WebSphere MQ queues and to CICS transactions.
- EP adapter emitting events over HTTP once APAR PK94205 is applied.
- SPI commands and CICS Explorer views for event bindings and capture specifications.

For more information, see the *What's New* section of the CICS TS V4.1 Information Center that has a topic on Support for event processing, which is available at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v4r1/topic/com.ibm.cics.ts.whatsnew.doc/ep/dfhe4_overview.html

1.7.2 CICS event capabilities in CICS TS V4.2

The initial event support was extended in CICS TS V4.2 to add the following capabilities:

- System events captured from a subset of state changes that can occur in the system, including DB2 connection status, file enable and open status, task threshold, tranclass task threshold, and transaction abends.
- Assured event emission with a new synchronous event emission mode, enabling events to be emitted in-line with the application from which the event was captured (in addition to the original asynchronous emission mode).
- New EP adapter resource that is independent of the event binding (embedded EP adapters also are still supported but are less flexible).
- New HTTP EP adapter to emit events to HTTP servers.
- EP Search capability to help identify potential effect of application changes on capture specifications.
- Extended SPI and CICS Explorer views to provide more detail about capture specifications and for EP adapters.
- Enhancements to the TS queue EP adapter to emit events to TS queues in XML formats.
- The custom EP adapter interface is passed the configuration in a character container for easier code page conversion.
- Event capture global user exit, XEPCAP.
- New support for capturing data in other data types, including floating point and COBOL zoned data types.

For more information, see the *What's New* section of the CICS TS V4.2 Information Center that has a topic on Improvements to event processing, which is available at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.ts.whatsnew.doc/ep/dfhe4_overview.html

1.7.3 CICS event capabilities in CICS TS V5.1

The event support was further extended in CICS TS V5.1 with the following other capabilities:

- New application capture point that allows events to be captured when the WRITE OPERATOR command is issued.
- New system event capture point that allows events to be captured when CICS messages are issued (for most CICS messages).
- New capture points when threshold policy rules are exceeded. For more information, see “CICS events and CICS threshold policies” on page 20.
- Static data can be defined in the event specification and included in the captured and emitted events.
- New EP adapter set resource to route one event to multiple EP adapters, and hence to emit to multiple event consumers.
- Updates to event emission advanced options.
- Security changes to the processing captured events.

For more information, see the *What's New* section of the CICS TS V5.1 Information Center that has a topic on Improvements to event processing, which is available at this website:

`http://pic.dhe.ibm.com/infocenter/cicsts/v5r1/com.ibm.cics.ts.whatsnew.doc/ep/dfhe4_overview.html`

CICS events and CICS threshold policies

CICS TS V5.1 introduces support for threshold policies. These policies provide a way of controlling resource usage. A policy rule defines the action to be taken when tasks that are running in a specified scope in a CICS platform exceed a certain threshold usage of a particular resource. One of the actions that can be specified when a task exceeds the policy threshold is to emit an event via a specified EP adapter.

CICS system events indicate that something interesting has happened in the CICS system. Threshold policies can be thought of as a way of specifying a predefined event that indicates that the usage of a resource has exceeded a certain threshold. The event action for a threshold policy causes an event to be emitted in exactly the same way as other system events in that the event is formatted and routed by the specified EP adapter. Therefore, events that are generated from threshold policies can drive other processing or be sent to other event consumers.

1.8 Summary

This chapter described the concepts and value of business events and event processing, reviewed some of the IBM products that can provide event processing solutions, and introduced the event processing support that is provided by CICS Transaction Server for z/OS. This support includes the capability to capture and emit application and system events.

2

Capturing application and system events

This chapter describes how CICS supports event processing and introduces the CICS Explorer editors that are used to define and deploy events.

The event binding editor is detailed to create an event specification, capture points, filters, information sources, and finally determines which EP adapter is to emit the event.

The CICS Explorer provides a set of example event bindings that are based on the CICS Catalog sample. One of these event bindings is used for illustration.

Finally, we describe how to use the CICS Explorer to deploy the event binding within a CICS bundle onto zFS.

This chapter includes the following topics:

- How CICS supports event processing
- CICS Explorer
- Event binding editor
- Explicit events by using SIGNAL EVENT
- Deploying a CICS bundle to zFS

2.1 How CICS supports event processing

CICS provides a flexible solution to capture application and system events, with options for filtering, enriching, and formatting the event, and emitting the event to one or more consumers. The events are typically used by event processing engines, business dashboards, monitoring tools, or CICS transactions, as shown in Figure 2-1.

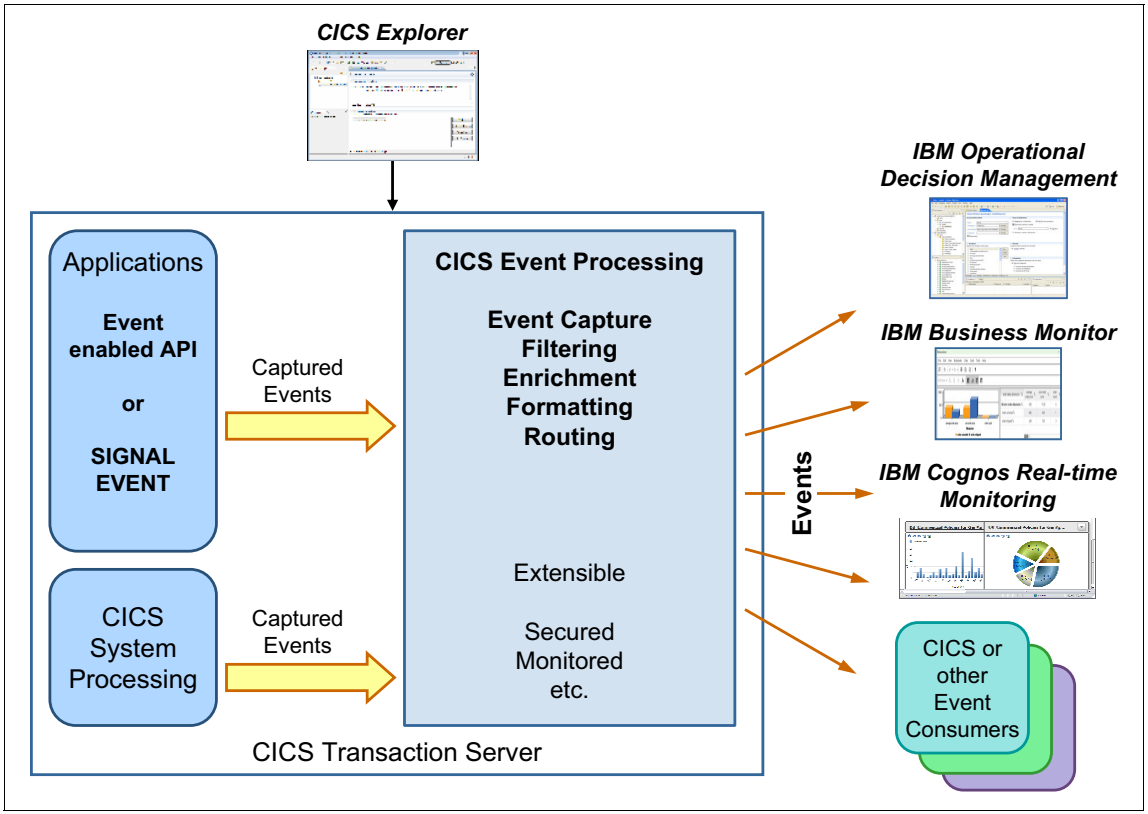


Figure 2-1 CICS and event processing overview

The CICS Explorer is used to create required resources and deploy them to CICS.

At a minimum, an event specification and an EP adapter specification are required. The event specification describes the information to be emitted, the point to capture the event, and the filters that are required to precisely capture only events of interest. The EP adapter specification describes which EP adapter should emit the event, how the data should be formatted, and advanced options such as priority and transactionally. If you require one event to be emitted to multiple EP adapters, you can also specify an EP adapter set specification.

It is recommended to create the event specification by using the CICS Explorer event binding editor and the EP adapter specifications by using the EP adapter editor. These specifications are created within a CICS bundle project, which can be exported as a CICS bundle onto the z/OS UNIX File System (zFS) and installed in CICS by using a CICS BUNDLE resource.

When CICS applications are run, many potential capture points occur. Possible capture points include when a program is initialized and when it runs one of the event-enabled CICS commands, such as REWRITE, LINK, and RECEIVE. These are referred to as *application events*. Applications can also declare capture points by using the SIGNAL EVENT command.

CICS also provides capture points for system activity, such as when the state of a DB2 connection changes, and when a workload reaches a percentage of a transaction class. These are referred to as *system events*.

For each EXEC CICS API that can be included as a capture point, CICS evaluates the event specifications that are installed and enabled. If there is a match, CICS captures the required information and enriches it with information about the application and system, and queues it for later processing. The application then continues without being aware the event was even captured.

Application events can be marked as transactional, in which case they are kept on the queue of captured events until the transaction within which the event was captured is successfully committed. Conversely, if the transaction is rolled back, the event is removed and not emitted.

A separate process takes events from the queue according to whether they have a normal or high priority order and routes them to the adapter that is specified in the event EP adapter specification. Each EP adapter formats the information to be included in the event (for example, into XML), and emits the event. This is referred to as *asynchronous event processing*.

CICS also supports synchronous event processing for application events where the EP adapter is called in-line when the event is captured from the application. This is useful if you want the application to be aware of errors when emitting events.

After an event is emitted, the event consumer (such as IBM Operation Decision Manager or IBM Business Monitor) typically update a dashboard or correlate it with other events and take some business action.

2.2 CICS Explorer

The CICS Explorer provides the following features to support event processing:

- A set of the following resource wizards and editors:
  - CICS Bundle Project
  - CICS Event Binding
  - CICS Event Processing Adapter
  - CICS Event Processing Adapter Set
  - Policy Definition
- A wizard to export CICS bundle projects to zFS.
- A facility to search event bindings in CICS bundle projects within the local workspace, or those installed into CICS systems.
- A set of the following CICS Operations views, including:
  - Event Processing
  - Event Bindings
  - EP Adapters
  - EP Adapter Sets
  - Bundles
  - Bundle Parts
- CICS SM definition views to create and manage BUNDLE resources.

The latest version of the CICS Explorer can be downloaded and installed from this website:

http://www.ibm.com/software/htp/cics/explorer/

2.2.1 CICS bundles

A CICS bundle is a collection of bundle parts, artifacts, references, and a manifest. Bundles can include version information and declare dependencies on other resources outside the bundle. Together, these make it easier to package, deploy, and manage applications, and enable CICS to identify problems and take corrective actions.

Complete the following steps to create a CICS bundle:

1. Start the CICS Explorer.
2. Switch to the Resource perspective by using the button as shown in Figure 2-2. Alternatively, select **Window** → **Open Perspective** → **Other** → **Resource** → **OK**.

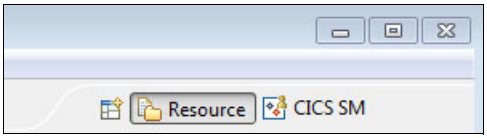


Figure 2-2 CICS Explorer Resource Perspective

3. Create a CICS Bundle project by using the icon that is shown in Figure 2-3. Alternatively, select **File** → **New Wizards** → **Other** → **CICS Resources** → **CICS Bundle Project**.

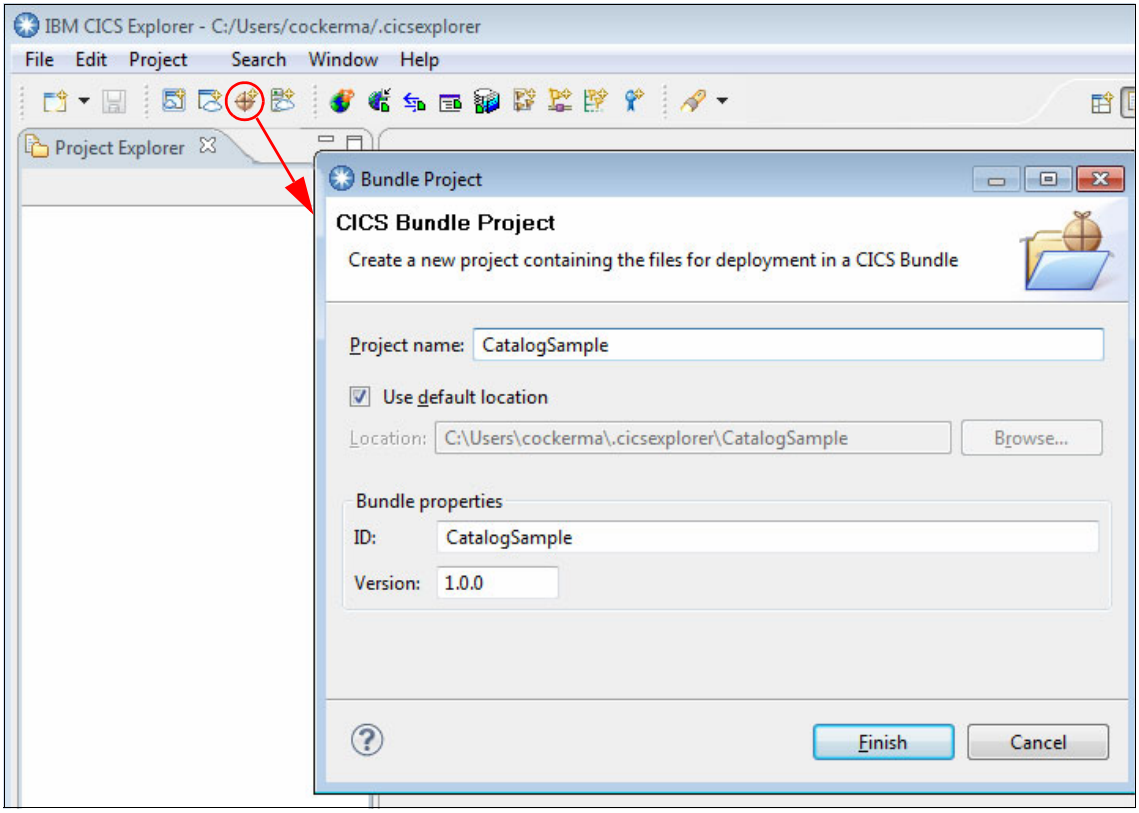


Figure 2-3 Creating a CICS Bundle project

4. Enter the project name `CatalogSample`.
5. Select **Finish**.

You are now ready to create event bindings, EP adapters, EP adapter sets, and policies in the CICS bundle.

2.3 Event binding editor

An event binding is an XML document that defines one or more events to CICS and consists of the following components, as shown in Figure 2-4:

- Event specifications
- Capture specifications
- An EP adapter specification, or a reference to an externally defined EP adapter or an EP adapter set

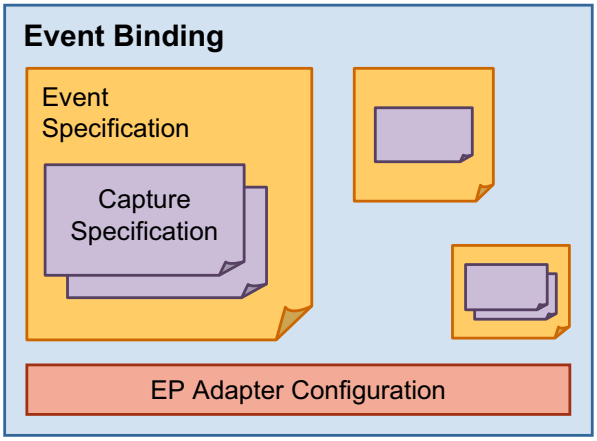


Figure 2-4 Event binding components

Complete the following steps to create the catalog example event binding:

1. Select the CatalogSample project.
2. Select **File** → **New Wizards** → **Other** → **Examples** → **CICS Catalog Manager REWRITE event**, as shown in Figure 2-5 on page 29.

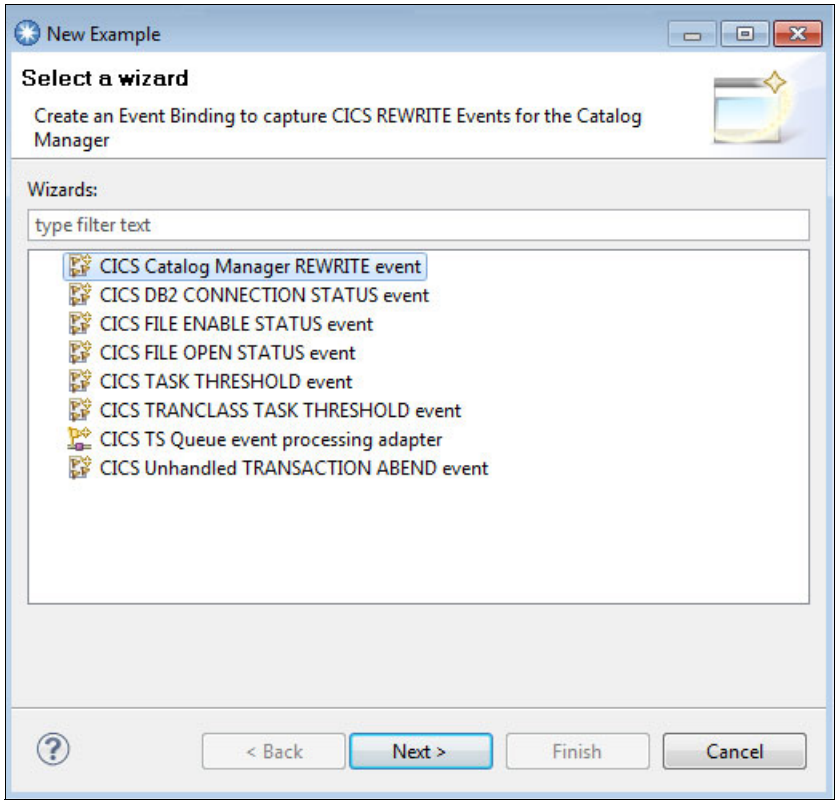


Figure 2-5 Generating a CICS Catalog Manager REWRITE event

3. Select **Next**.
4. Enter the file name `CatalogSample`, then select **Finish**.

The example is created and the event binding editor is started.

2.3.1 Event Binding tab

The event binding tab is divided into the following sections, as shown in Figure 2-6 on page 30:

- General Information: This section includes a description of the event in high-level business terms and a usertag to describe a version number.
- Event Specifications: This section lists the event specifications that are included in this event binding and provides controls to create, edit, remove, and copy specifications.

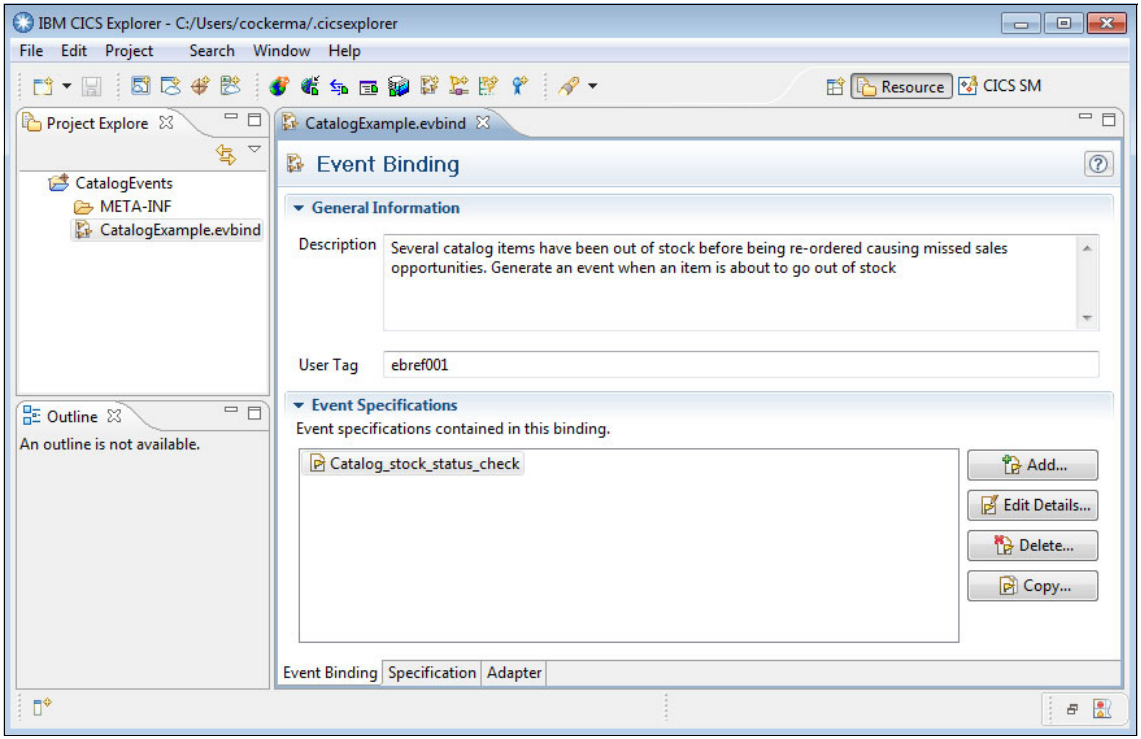


Figure 2-6 Catalog sample event binding

To display the Specifications tab, select the Catalog_stock_status_check event specification, then **Edit Details**.

2.3.2 Specifications tab

Each event specification includes one or more capture specifications and a set of the business information items to be emitted, as shown in Figure 2-7.

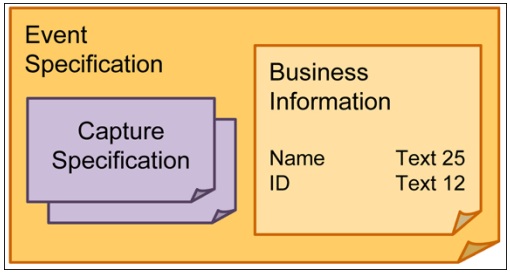


Figure 2-7 Event Specification components

Figure 2-8 on page 32 shows the event specification for the catalog sample, which is divided into the following sections:

- General: This section includes a description of the event specification.
- Emitted Business Information: This section lists the items to be included in the event, in the order they are to be included. Each item includes a data type, length, and numeric precision. As shown in Figure 2-8 on page 32, the emitted items include Program_name, Item_ref, Item_description, in_stock, and on_order.
- Capture Specifications: This section includes a button that is used to add new capture specifications. Alternatively, you can right-click the event specification in the tree on the left side and select **Add a Capture Specification** in the menu.
- Automatic Capture Specification: The use of automatic capture specifications is described in 2.4, “Explicit events by using SIGNAL EVENT” on page 42.

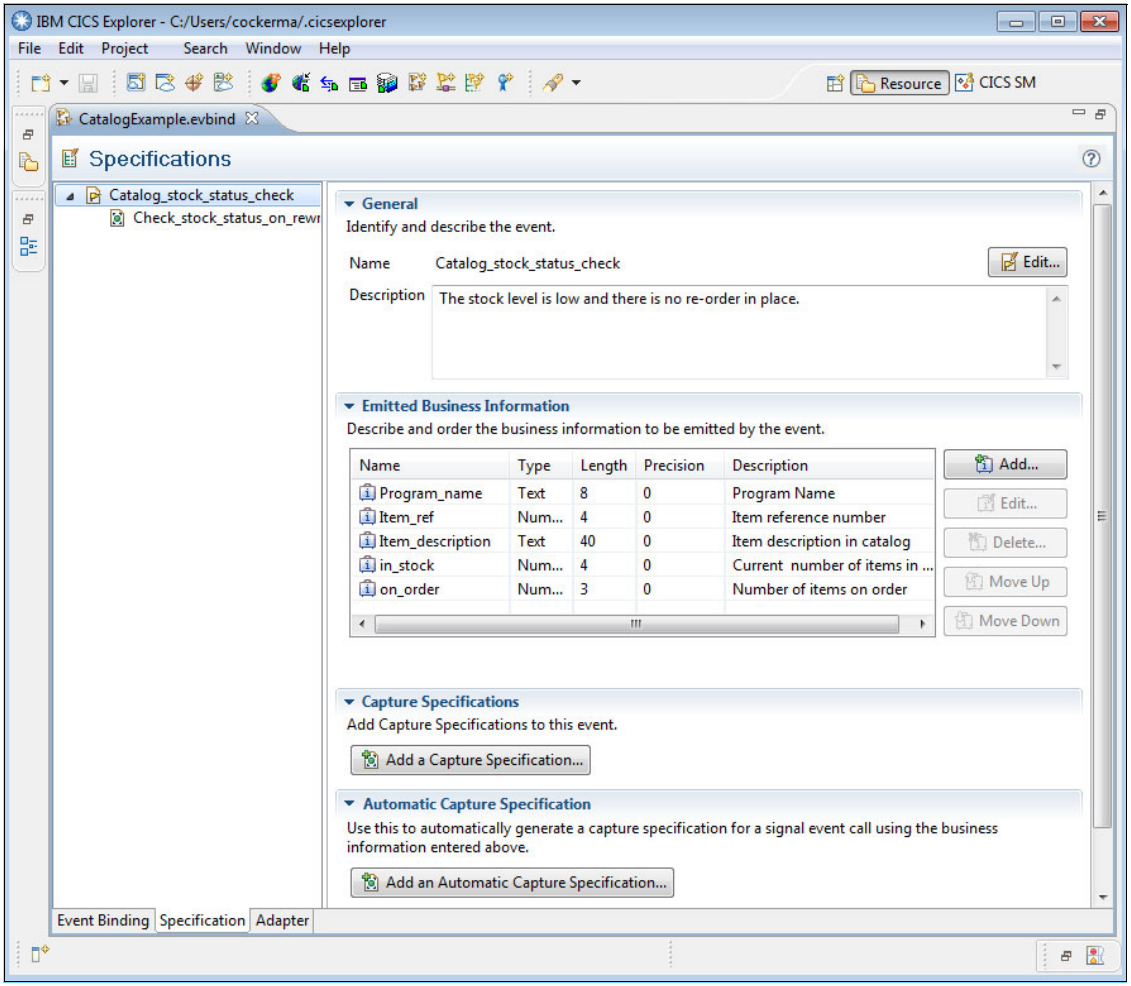


Figure 2-8 Catalog sample event specification

2.3.3 Capture Point tab

A capture specification defines the place in CICS where a particular event can be captured. It consists of a capture point, filter predicates, and information sources for the data to be captured from, as shown in Figure 2-9.

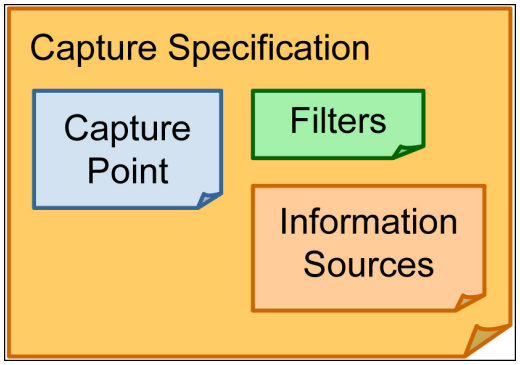


Figure 2-9 Components of a capture specification

If there is more than one place in the application where the event occurs, you can add a capture specification for each scenario. For example, a stock application can provide an entry point for orders to be placed via a web service and another for mobile applications. The capture points and data formats are likely to be different; therefore, two capture specifications are needed.

Capture point

A capture point is an event-enabled CICS command that is run by an application or a stage in CICS processing that was event-enabled. Table 2-1 lists application-related capture points.

Table 2-1 Application capture points

CONVERSE[a]	DELETE FILE	DELETEQ TD	DELETEQ TS
INVOKE SERVICE[a]	LINK PROGRAM[a]	PROGRAM INIT[b]	PUT CONTAINER
READ	READNEXT	READPREV	READQ TD
READQ TS	RECEIVE	RECEIVE MAP	RETRIEVE
RETURN[b]	REWRITE	SEND	SEND MAP
SEND TEXT	SIGNAL EVENT	START	WEB READ
WEB READNEXT	WRITE FILE	WRITE OPERATOR	WRITEQ TD
WRITEQ TS	XCTL[b]		

a. CONVERSE, INVOKE SERVICE, and LINK PROGRAM capture points can be evaluated before or after the command runs.
b. PROGRAM INIT, RETURN, and XCTL capture points are evaluated before the CICS command runs.

For more information about cross-reference of capture points and their corresponding filter predicates and information items, see Appendix , “CICS application capture points” on page 240.

Capture points are evaluated after the CICS command runs, except where noted.

Table 2-2 lists the CICS system-related capture points.

Table 2-2 System capture points

DB2 CONNECTION STATUS	FILE ENABLE STATUS	FILE OPEN STATUS
MESSAGE	TRANSACTION ABEND	TASK THRESHOLD
TRANSCLASS THRESHOLD		

Select the Check_stock_status_on_rewrite capture point from the tree in Figure 2-8 on page 32 to see the catalog sample capture point, as shown in Figure 2-10 on page 35. The capture point for this sample event is REWRITE.

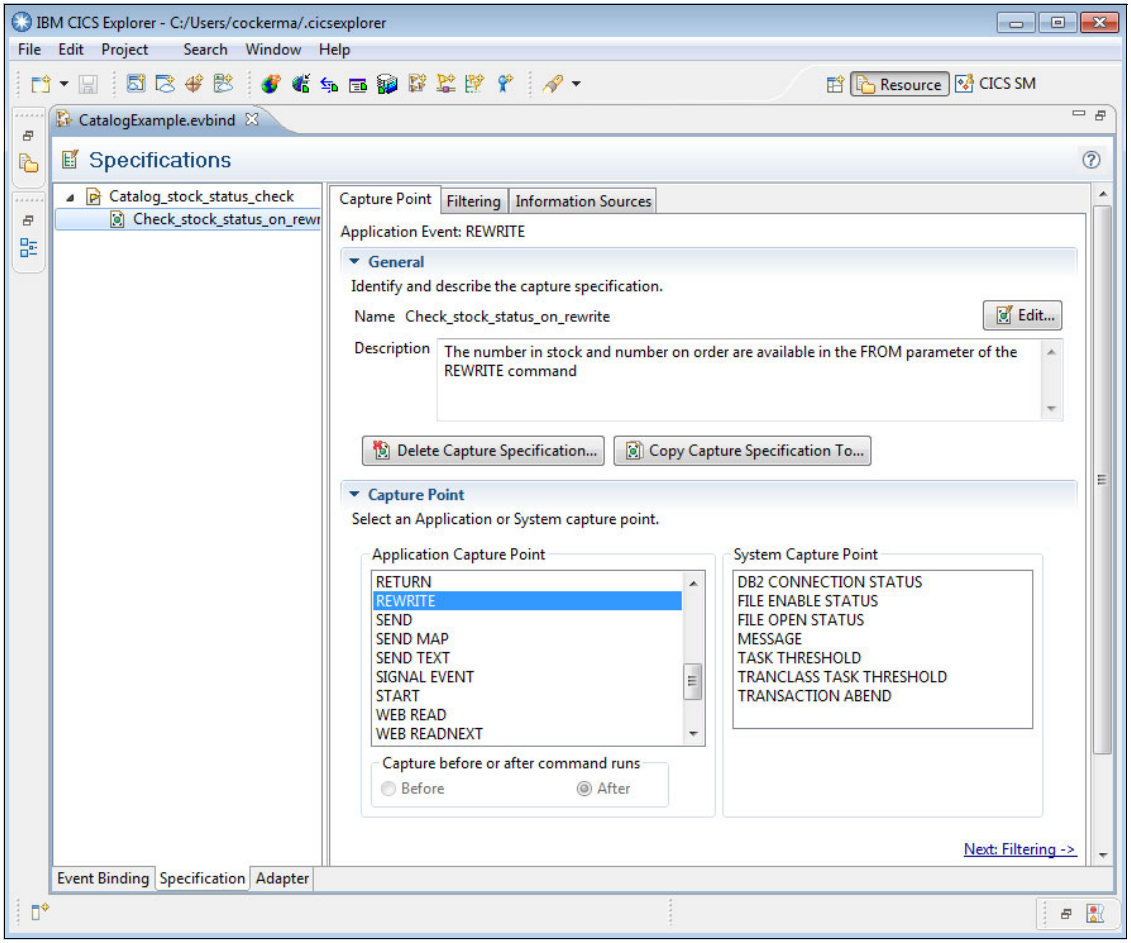


Figure 2-10 Catalog sample REWRITE capture point

2.3.4 Filtering tab

A predicate is an expression that is used as part of a filter, which consists of a data item, an operator such as Equals or Starts With, and optionally a value. A predicate is evaluated against data values on the capture point command or context data.

A filter is a set of predicates. If the predicates in the filter all evaluate to true, CICS captures the event. Conversely, if any predicates evaluate to false, CICS does not continue to evaluate the remaining predicates and the event is not captured.

Select the Filtering tab to show the catalog sample filter predicates, as shown in Figure 2-11.

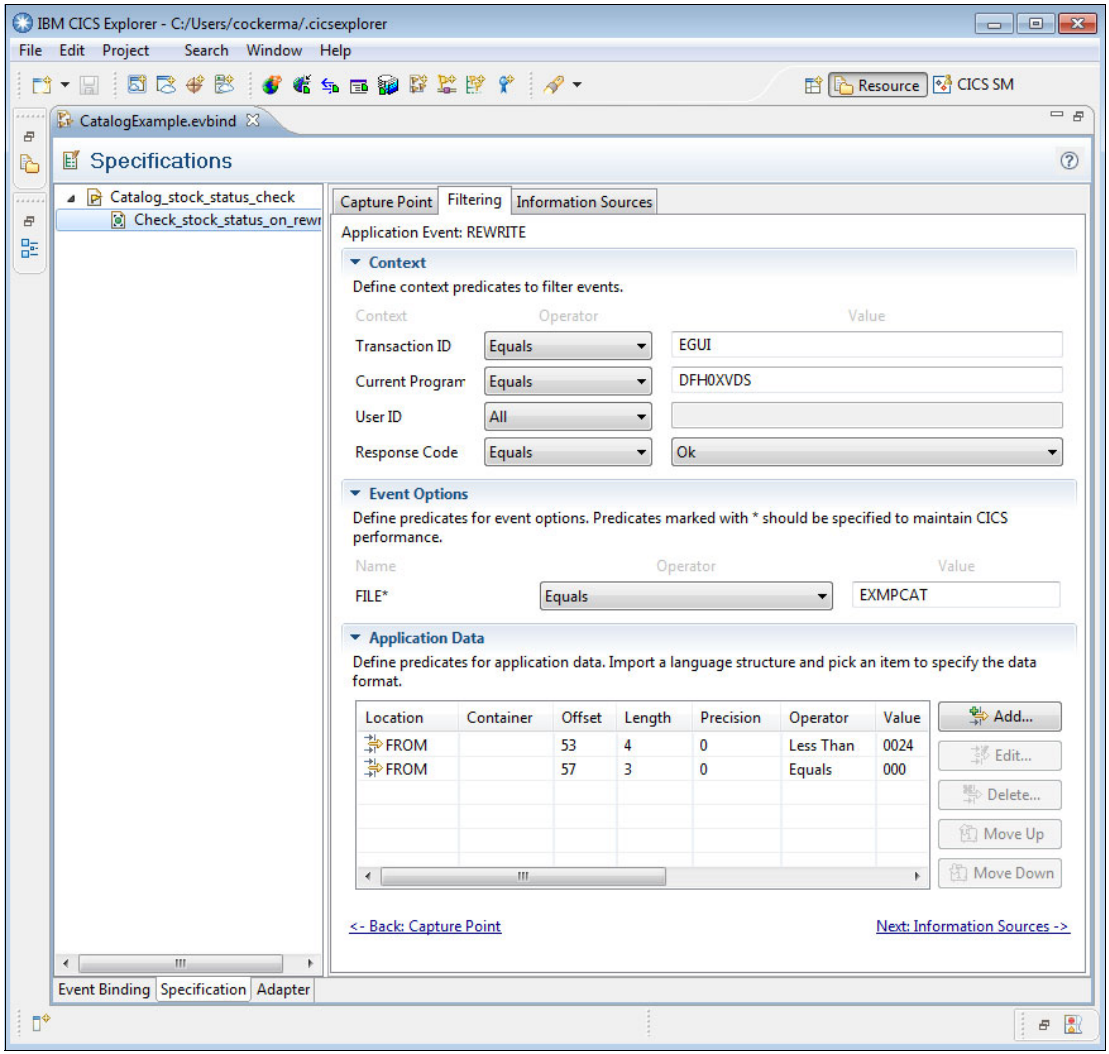


Figure 2-11 Catalog sample filtering

The Filtering tab is divided into the following sections:

- Context: These predicates enable filtering that is based on transaction ID, current program, user ID, and the CICS command response code. The last of these are relevant only for capture points that are evaluated after the command has run.

- Event Options: These predicates are different for each capture point. For the REWRITE command capture point that is used here, the predicate is for the file name.
- Application Data: These predicates are different depending on the capture point. For the REWRITE command capture point, you can define multiple predicates for the data that is passed by the program on the FROM option when it issued the REWRITE command.

Getting the right combinations of predicates in the filter is important to ensure that the required events always are captured and, for efficiency, unwanted events are not captured.

Tip: If you need a combination of predicates that is too complex or not possible to express by using the editor, you must modify the application to perform those tests and when true issue the SIGNAL EVENT command. The SIGNAL EVENT capture point then can be used.

2.3.5 Information Sources tab

Complete the following steps if you want the event to contain specific data:

1. Add business information items to the event specification in the order you want them emitted.
2. Define an information source for each item of business information in the capture specifications for the event.

   Figure 2-12 on page 38 shows information sources for the catalog sample.

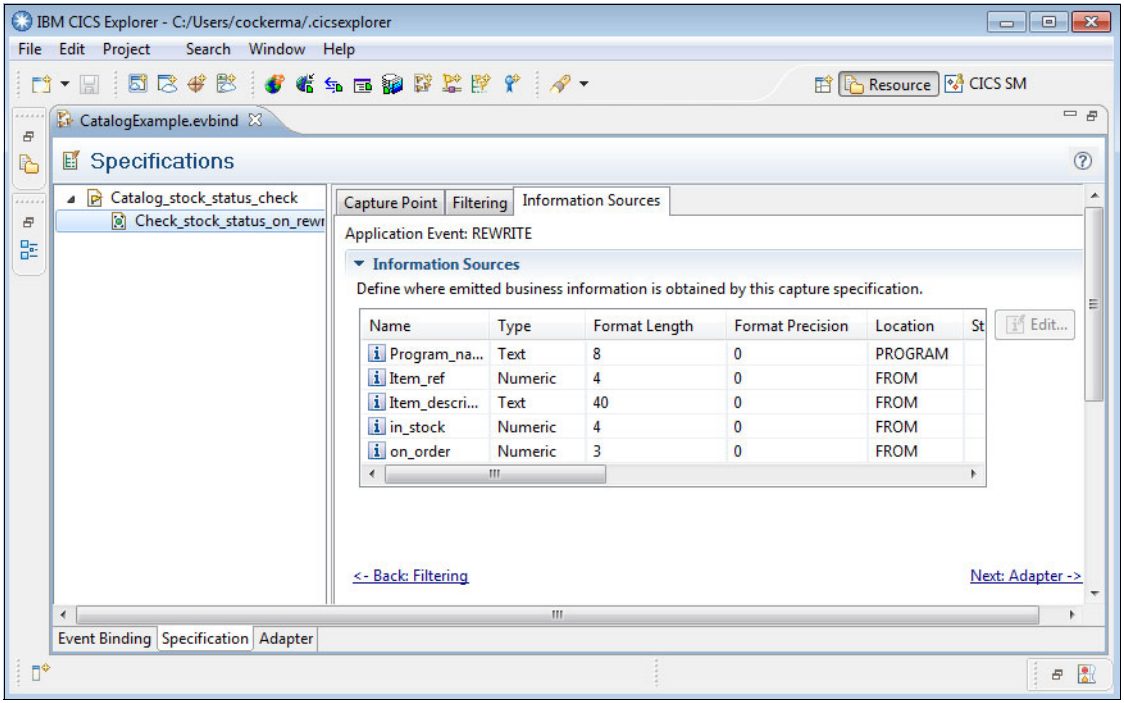


Figure 2-12 Catalog sample information sources

3. In the Information Sources section, select the `Item_description` item, then click **Edit**. Figure 2-13 on page 39 shows the information source editor.

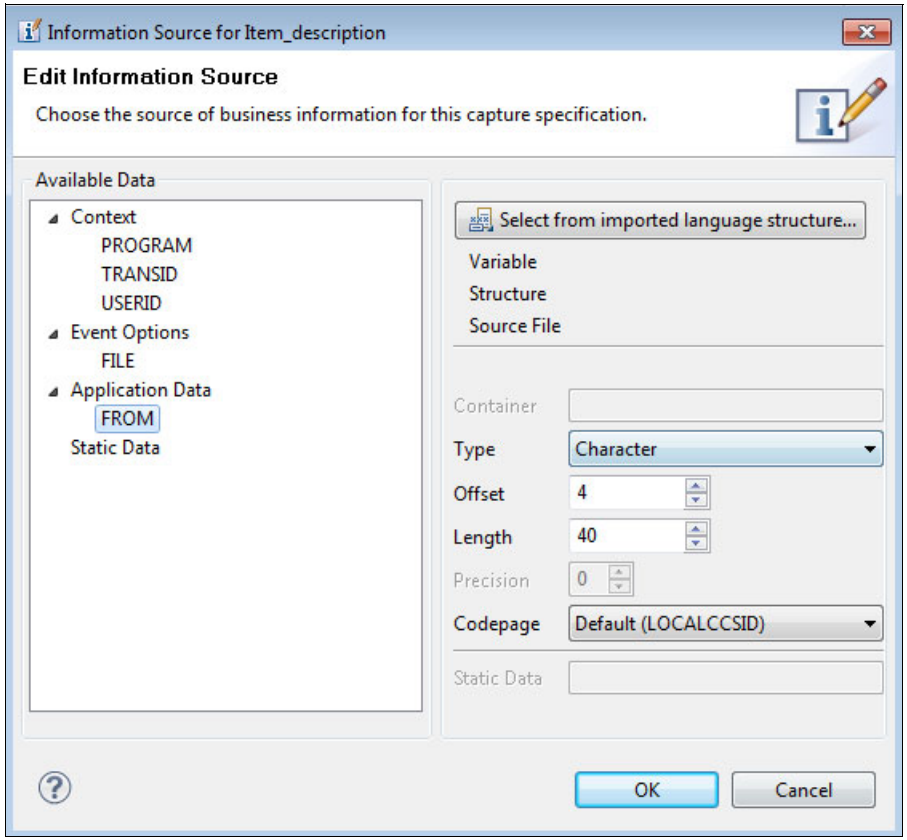


Figure 2-13 Editing an information source

The content of this editor is relevant to the capture point; in this case, the REWRITE application capture point.

The available data is divided into the following areas:

– Context

  These are provided by CICS, and for application capture points including USERID, PROGRAM, and TRANSID.

– Event options

  These are provided by the application on the application command.

– Application data

  These are provided by the application on the application command options. You must provide extra information, such as the container name, type, offset, length, numeric precision, and code page.

If you have an application language structure that defines the necessary fields (such as a COBOL copy book), use the **Select from imported language structure** button to import and select the field.

– Static data

This is a way of providing data that is to be statically included each time the event is emitted, and is not based on the value of any data at run time.

This is provided in the Static Data field on the right side of the GUI.

Tip: If you need the event to contain information that is not supplied by an existing capture point, you must modify the application to issue the SIGNAL EVENT command and supply the data in the FROMCHANNEL or FROM options. The SIGNAL EVENT capture point can then be used.

4. Select **OK**.
5. Select the Adapter tab.

2.3.6 Adapter tab

The Adapter tab is used to specify which EP adapter or EP adapter set should be used to emit the event, or to define the EP adapter specification within the event binding.

The tab is divided into the following sections:

- Resource: This section is used to select if the event should be emitted by using the EP adapter specification that is defined in an EPADAPTER resource, an EPADAPTERSET resource, or defined in the Adapters section.

  Use the Export Event Specification button to create two files that define the data in the emitted event: as an XML scheme in a `.xsd` file, and as COBOL copy book in a `.cpy` file. Only one of these files is required, depending on the Data Format option that is selected in the EP adapter specification.

- Adapters: This section is used to select the EP adapter to be used to emit the event with configuration that is appropriate for that adapter. For example, the TS Queue adapter requires the queue name, queue system identifier if the queue is remote, if the queue should use auxiliary storage, and the data format. For more information, see Chapter 3, “Event emission” on page 49.

- Advanced Options: This section is used to select the emission mode, dispatch priority, transaction ID, user ID, and if events are to be transactional. For more information, see Chapter 3, “Event emission” on page 49.

Tip: It is recommended to use an EPADAPTER or EPADAPTERSET resource because it is likely that several event specifications can use the same EP adapter specification. Also, any changes to the EP adapter specification do not require event bindings to be modified.

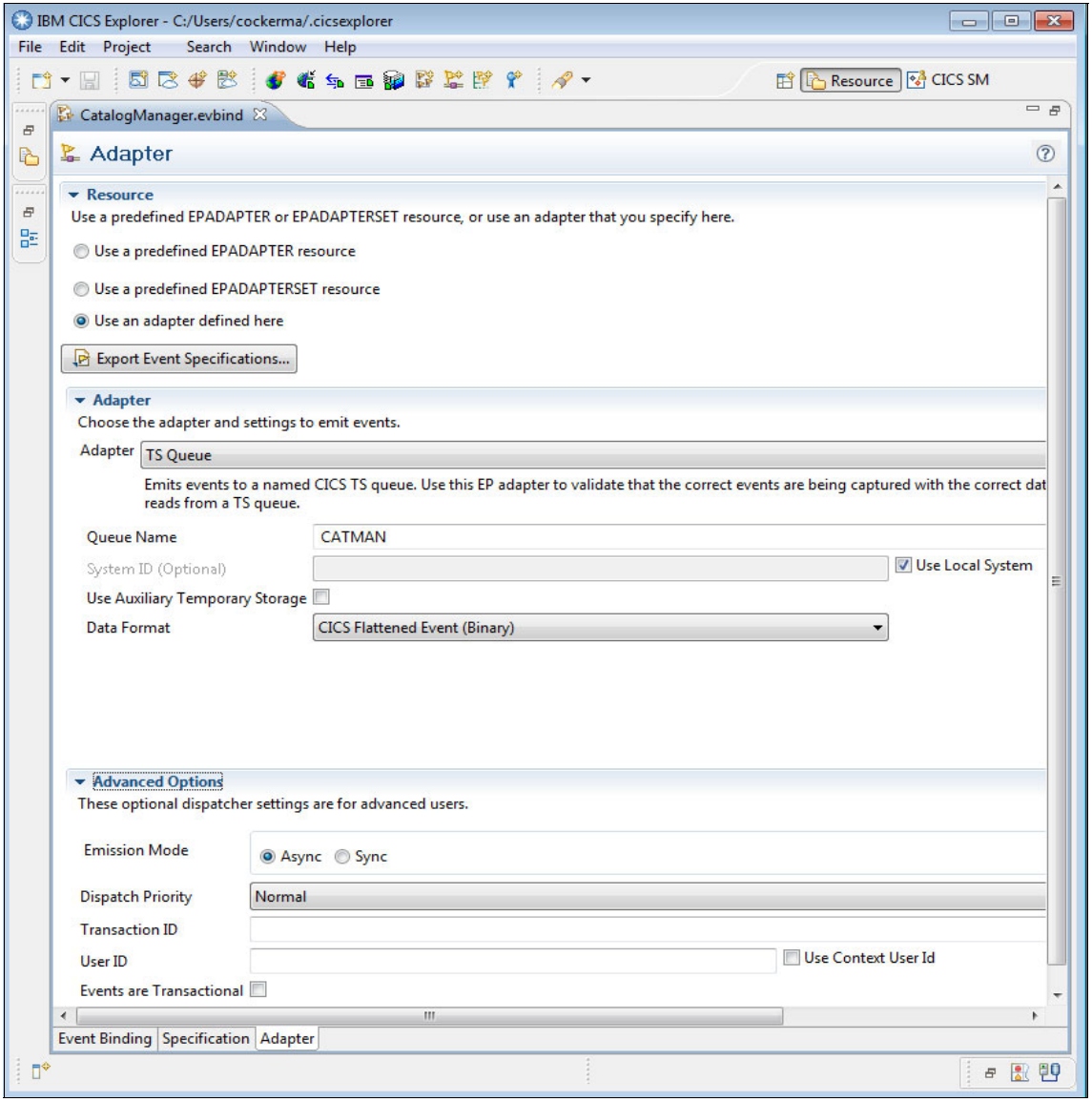


Figure 2-14 EP adapter information for the catalog sample

2.4 Explicit events by using SIGNAL EVENT

In addition to capture points for commands that are already in use in your applications, CICS enables applications to identify specific capture points by using the SIGNAL EVENT command.

The format of the command is shown in Example 2-1.

Example 2-1 SIGNAL EVENT command format

```
>>-SIGNAL EVENT(data-value)--+-------------------------------------------+-><
                             +-FROMCHANNEL(data-value)-------------------+
                             '-FROM(data-area)-+-----------------------+-'
                                               '-FROMLENGTH(data-value)-'
```

The command includes the following elements:

- Event(data-value) specifies the event name
- FROMCHANNEL specifies the name of a channel that has containers from which event information can be captured
- FROM provides a storage area from which event information can be captured

SIGNAL EVENT works in the same way as other capture point in CICS in that an event specification and capture point are required. However, the following differences are notable:

- You can explicitly choose where in the application logic the capture point occurs, rather than relying on the use of an existing command.

  This ability is useful where there is no suitable existing command that is event-enabled (such as an SQL command) or complex logic is required to establish if the event is of interest that cannot be expressed by using the event binding editor.

- You provide a name for this capture point.

  This function can make it easier to capture events without requiring the user of the event binding editor to have a deep knowledge of the application code. In particular, this might be the case if the application is provided by an IBM Business Partner and the application source is not available.

  Capture points can filter on this name with operators Equals or Begins With.

- You can assemble the data you want to make available from diverse sources (such as DB2 tables) that are not available on event-enabled commands.

  A capture specification for this SIGNAL EVENT can capture information sources from this data to build the event payload.

Although SIGNAL EVENT requires the application to be changed, if a good selection of data is included on the command, this change must be made only once. After that change is made, an event specification can be used to determine whether the event is enabled and what data is emitted.

2.4.1 Automatic Capture Specification for SIGNAL EVENT

If you use the SIGNAL EVENT command that uses FROMCHANNEL and put the data into containers, you can use the editor to create a capture specification for the event automatically.

Complete the following steps to add an automatic capture specification by using the event binding editor:

1. Add a new event specification with a name that matches the `EVENT` predicate.
2. Add the business data you want to capture with the names of the containers you are passing on the `FROMCHANNEL`.
3. Select the **Add an Automatic Capture Specification** button, as shown in Figure 2-8 on page 32.

The editor creates a capture specification for a SIGNAL EVENT capture point with the filter predicate as shown in the following example:

```
EVENT Equals <name of event specification>
```

It also adds information sources for each business data and includes the following parameters:

- The information is in a container with the name of the business data that is passed on the FROMCHANNEL.
- The data is at offset 0 in the container.
- The length is the same as the length of the business data.

Figure 2-15 shows an event specification to be used with SIGNAL event.

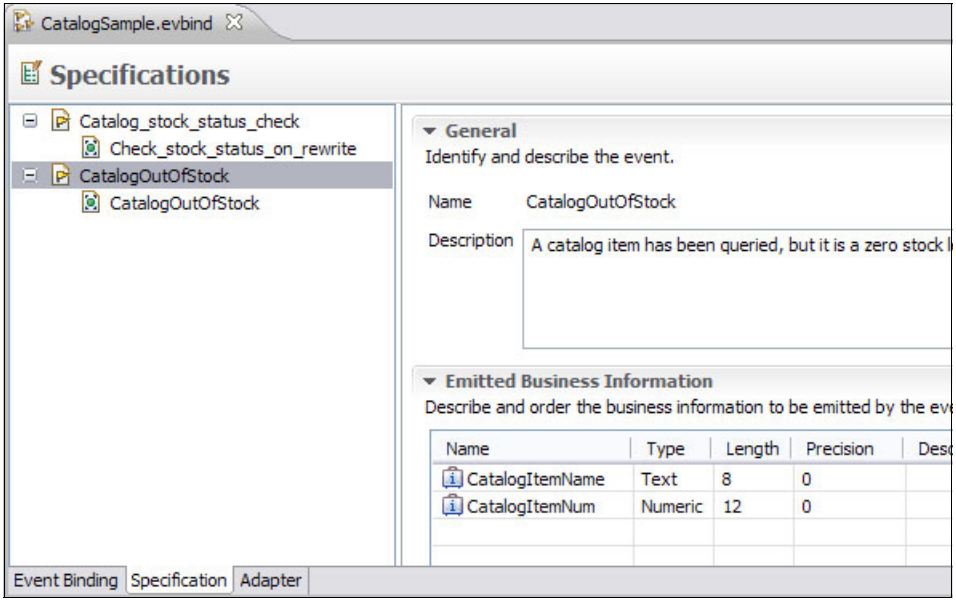


Figure 2-15 Event specification for a SIGNAL EVENT

Figure 2-16 on page 45 shows an automatically-generated capture specification.

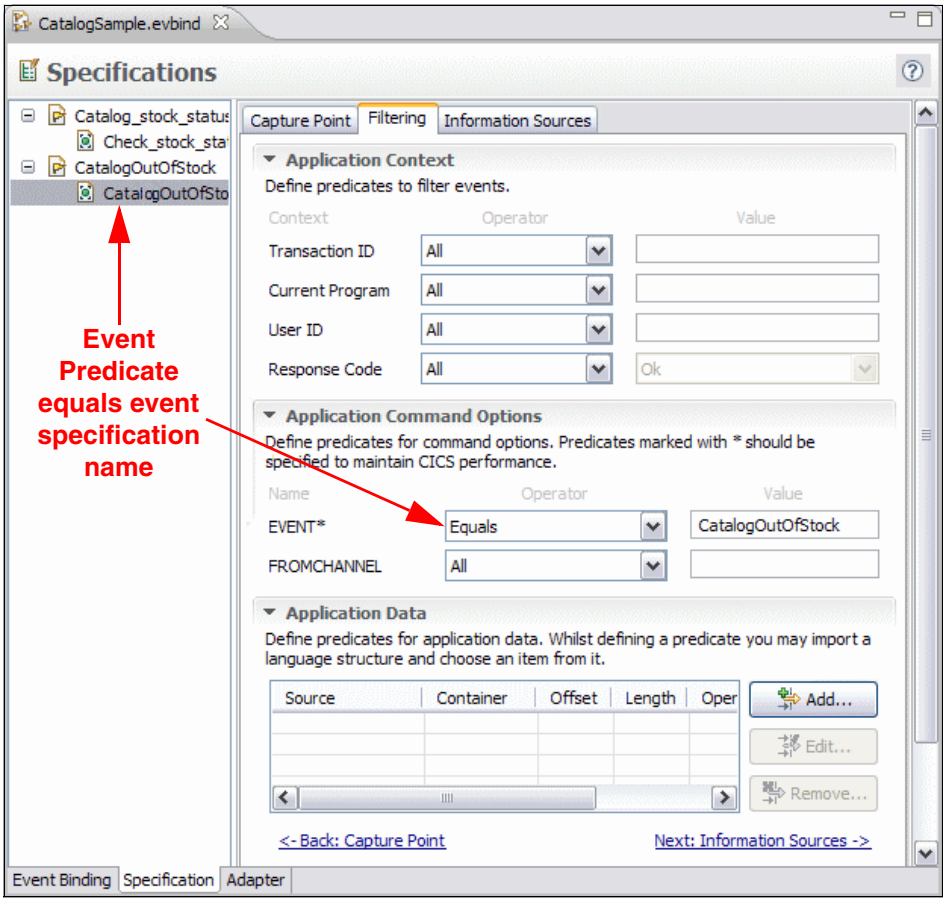


Figure 2-16 Automatically generated capture specification

Figure 2-17 shows the automatically-generated information sources in the capture specification.

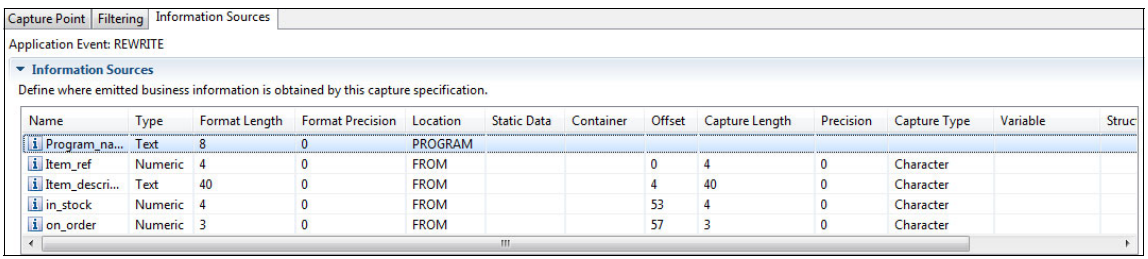


Figure 2-17 Automatically generated information sources

2.5 Deploying a CICS bundle to zFS

The CICS Explorer provides an export wizard to deploy a CICS bundle project to a zFS directory, which is ready for it to be installed into CICS by using a BUNDLE resource.

Complete the following steps to deploy a CICS bundle to zFS:

1. To start the export wizard, right-click the CICS bundle project and select **Export Bundle Project to z/OS UNIX File System**, as shown in Figure 2-18.

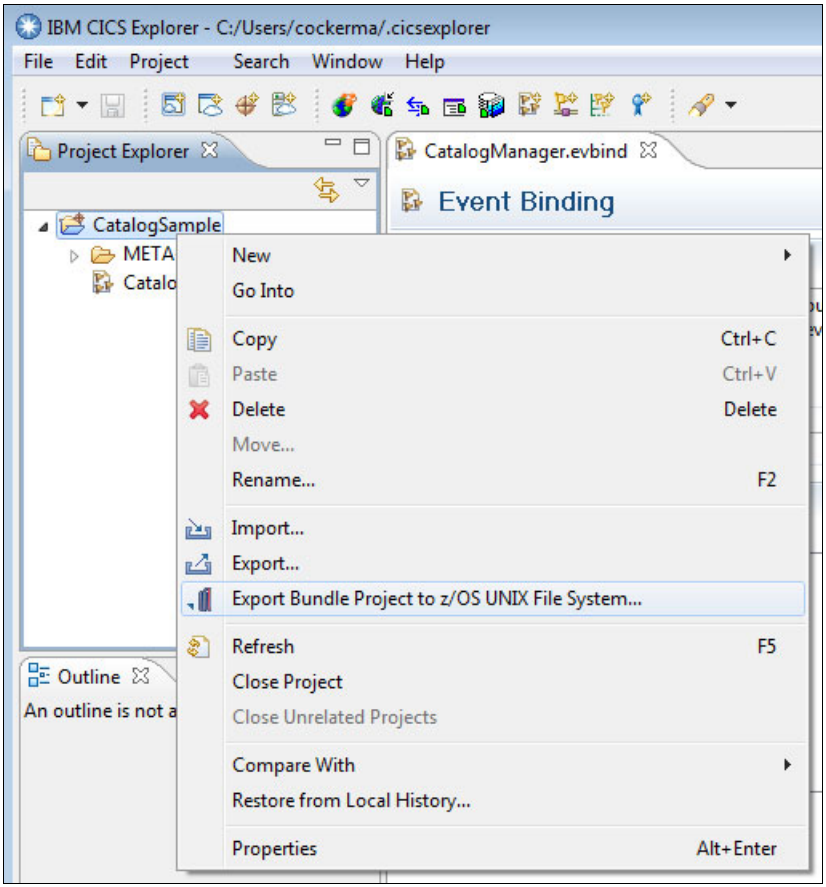


Figure 2-18 Selecting the export panel

2. Select **Export to a specific location in the file system** → **Next** and complete the information in the Export Bundle panel to provide the bundle project name, connection, and directory details.

It is advisable to select the **Clear existing contents of Bundle directory** option if the directory already exists to remove any pre-existing files, as shown in Figure 2-19.

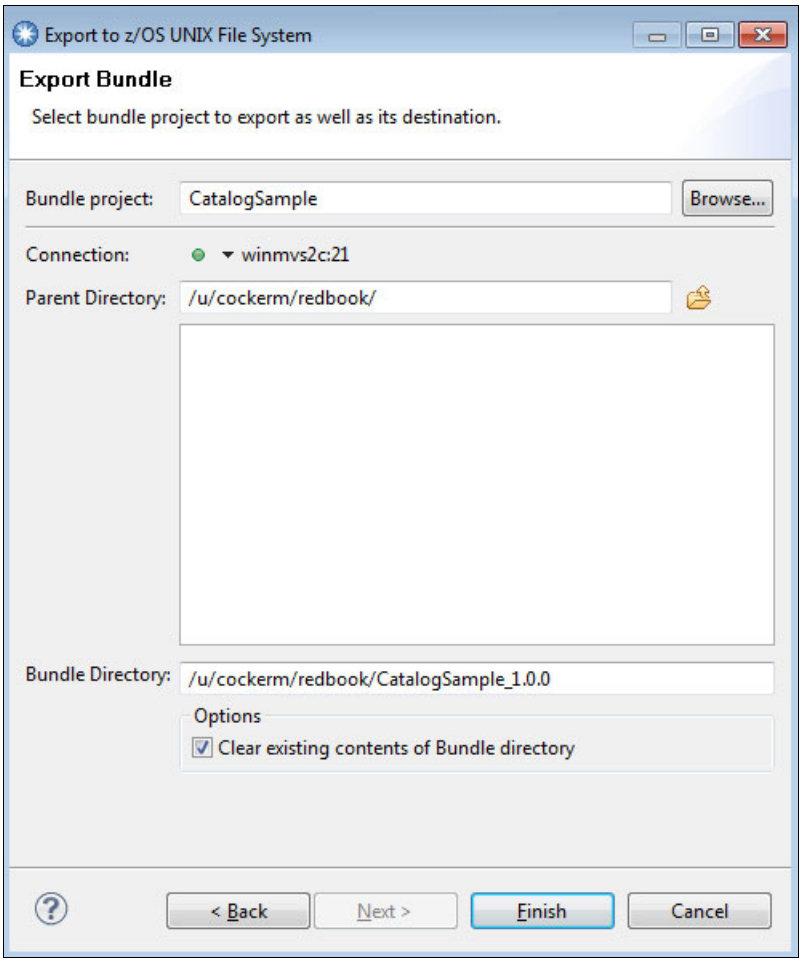


Figure 2-19 Export bundle panel

3. Select **Finish**. The CICS bundle project is exported.

3

Event emission

In this chapter, we describe event emission, including the different kinds of event processing (EP) adapters that emit events as different formats to different types of destination. The options to control event emission also are described, the EP adapter set is introduced, and exporting event specification is described.

This chapter includes the following topics:

- Event processing adapters
- EP Adapter advanced options
- Predefined EP adapters
- EP adapter set
- Exporting event specifications

3.1 Event processing adapters

EP adapters are programs that format and then emit events from a CICS system. An event binding needs an EP adapter to be specified for the event to be emitted by CICS.

The following EP adapters are provided by EP:

- TS queue EP adapter: Easy to set up for use within CICS, but has a limit of 32,767 entries in a TS queue.
- Transaction start EP adapter: Emits events to start CICS transaction.
- WebSphere MQ EP adapter: Uses a robust transport mechanism and provides a quick and easy way of emitting events out of CICS.
- HTTP EP adapter: Provides a quick and easy way of emitting events out of CICS.
- Custom EP adapter: If none of these adapters serve your purpose, you can provide a CICS program as EP adapter to format and emit the events.

If you want the event to be transactional and synchronous, WebSphere MQ and TS queue EP adapters are good choice because they can emit events in a recoverable way.

You can specify an EP adapter with a definition embedded in an event binding, or specify one which is predefined and thus is separate from the event binding. The advantage of specifying a predefined EP adapter is the EP adapter can be used in more than one event bindings. Predefined EP adapters can be included in an EP adapter set, which is described in 3.4, “EP adapter set” on page 60. Despite the name, a predefined EP adapter does not need to be defined when it is specified in the event binding or EP adapter set. However, it does need to be defined and installed before events can be emitted.

To view an EP adapter definition, open the catalog sample event binding in CICS Explorer and select the Adapter tab in the CICS event binding editor. The adapter pane is where you specify the EP adapter to emit events that are created by this event binding. You can specify the parameters for the EP adapter and any advanced information.

Select the type of EP adapter, then select options relevant to the EP adapter. Figure 3-1 shows the adapter configuration for the catalog sample event binding.

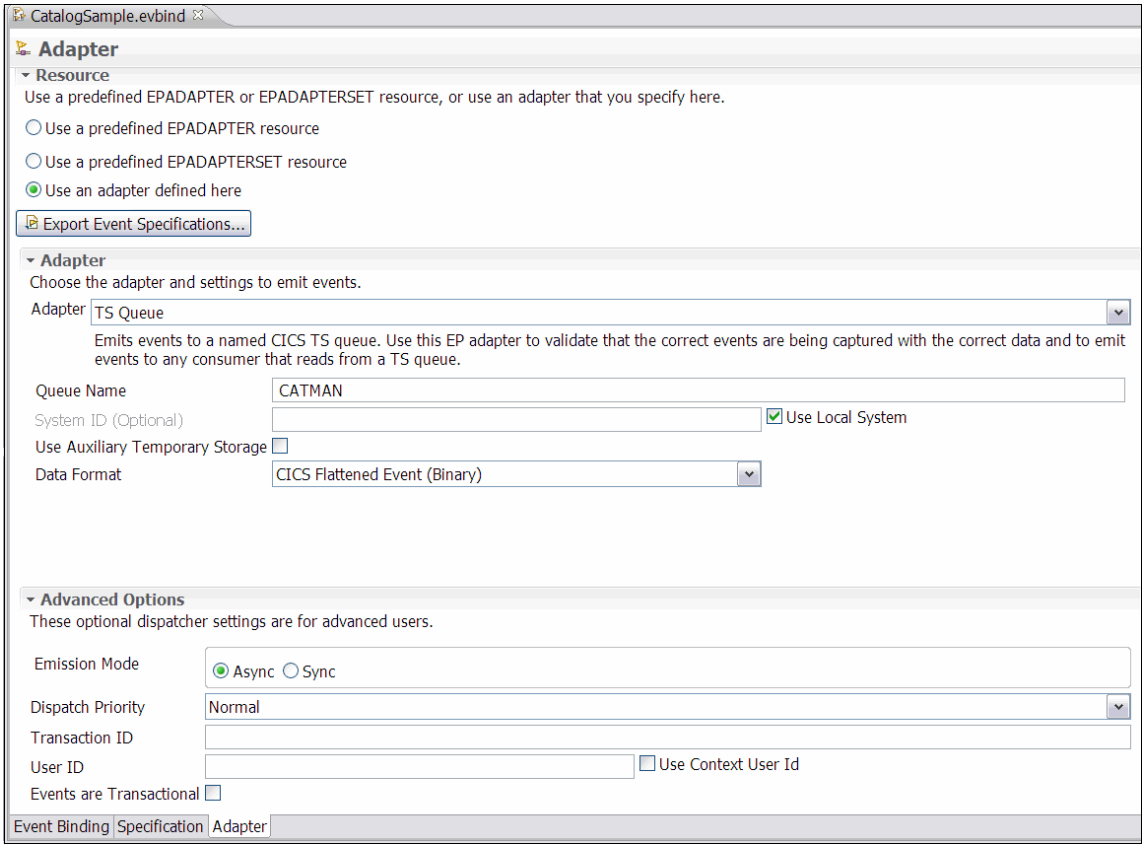


Figure 3-1 Adapter information for the catalog sample

Use the drop-down menu for the Adapter field to choose an EP adapter type. EP adapter types are described in 3.1.1, “TS Queue EP adapter” on page 52 through 3.1.5, “Custom EP adapter” on page 54. The event formats supported by each type of EP adapter are described in 3.1.6, “Data formats supported by EP adapters” on page 54.

You can export a description of your event as an XML schema or COBOL copy book. For more information, see 3.5, “Exporting event specifications” on page 62.

Optionally, you can set advanced dispatcher settings to control the behavior of EP adapter. For more information, see 3.2, “EP Adapter advanced options” on page 55.

3.1.1 TS Queue EP adapter

This EP adapter emits events to a named CICS temporary storage queue and can be used to perform the following tasks:

- Validate that the correct events are being captured with the correct data.
- Emit events to any consumer that reads from a TS queue.

This EP adapter is a good choice when events are implemented. You can use it for testing and debugging purposes because you can browse TS queues with the CICS-supplied CEBR transaction to check the payload. You can also refresh the view to check for new events in the queue.

However, TS Queues are limited to 32,767 entries or less. You might delete only the entire queue rather than individual entries within, so you need a strategy for clearing out events you processed.

Specify the following options for the TS queue EP adapter:

- The CICS queue name.
- The System ID if your target queue is remote.
- Select **Use Auxiliary Temporary Storage** if required and if the TS queue does not already exist.
- A data format for the event. For more information, see “Data formats supported by EP adapters” on page 54.

3.1.2 Transaction start EP adapter

This EP adapter emits events in the CICS container-based event (CCE) format to start a named CICS transaction that uses the event data. You can specify the CICS system that runs the transaction.

Specify the following options for the Transaction start EP adapter:

- The transaction ID of the CICS application that runs as a result of the events.
- Optionally, a transaction user ID or select **Use Context User ID** for the transaction to run under. If none of the IDs are specified, the started transaction runs under the CICS default user ID.

If you specify a transaction user ID, CICS checks that the installation user ID is authorized as a surrogate user of the transaction user ID when the adapter is installed.

3.1.3 WebSphere MQ EP adapter

This EP adapter emits events to a WebSphere MQ queue in an XML format for use by products that use the common event infrastructure and Common Base Event format, such as IBM Business Monitor, WebSphere Business Event format for IBM Operational Decision Manager, or in a non-XML character format.

Specify the following options for the WebSphere MQ Queue 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 adapter are nonpersistent.
  - Yes: Messages put on the queue by the adapter are persistent.
  - Queue default: Messages put on the queue inherit the default persistence of the named queue.
- Specify the message priority. You can select the queue default, or enter a value (0 - 9) in the Priority field for the WebSphere MQ message priority.
- Specify the expiry time. You can select Never Expire or enter a value for the WebSphere MQ message expiry in the Expiry Time field. This period is expressed in tenths of a second. A message can be discarded if it was not removed from the destination queue before this period elapses.
- Specify a data format for the event.

3.1.4 HTTP EP adapter

This EP adapter emits events to an HTTP 1.1-compliant server by using HTTP POST in XML format through the connection that is described in a URIMAP with USAGE(CLIENT). It can be used as a less robust alternative to WebSphere MQ.

Specify the following options:

- The URIMAP name that represents the connection to the HTTP server. You must specify this option.
- A data format for the event.

3.1.5 Custom EP adapter

A custom (user-written) EP adapter 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 CICS-provided EP adapters. It must not carry out any other processing, such as usage of the event.

Specify the transaction ID for your user-written CICS application that formats, routes, and emits the event. You must specify a transaction ID.

Write the data to be passed to the Custom EP adapter. Your custom EP adapter receives this data that can be used to configure the custom EP adapter

3.1.6 Data formats supported by EP adapters

Table 3-1 lists the data formats that are supported by different types of EP adapters.

Table 3-1 Data formats supported by EP adapters

TS queue EP adapter	HTTP EP adapter	Transaction start EP adapter	WebSphere MQ EP adapter
► Common Base Event ► Common Base Event REST ► WebSphere Business Events ► CFE	► Common Base Event ► Common Base Event REST ► WebSphere Business Events	CCE	► Common Base Event ► WebSphere Business Events ► CFE

The following abbreviations of and terms for the event formats are used in Table 3-1:

- Common Base Event

  Event data is the Common Base Event XML format that is required by IBM Business Monitor.

- Common Base Event REST format

  Event data is the Common Base Event XML format without the Common Base Event envelope.

- WebSphere Base Event

  Event data is the XML format that is required by WebSphere Business Events.

- CICS flattened event (CFE)

  Event data is in a COBOL language structure.
- CICS container-based event (CCE)

  Event data is placed in containers in a CICS channel named DFHEP.EVENT.

3.2 EP Adapter advanced options

Advanced dispatcher options are shown in Figure 3-2. These options are for advanced users and control the way in which the EP adapter is run in a CICS system. They are not required.

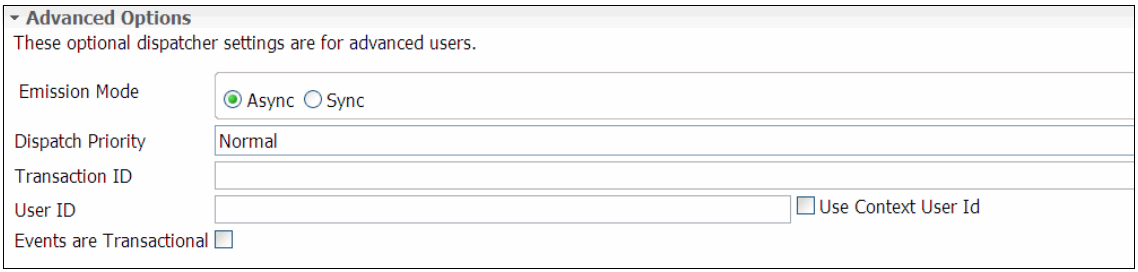


Figure 3-2 Advanced options for dispatcher

It is generally more efficient to allow 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 must 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. In these cases, you must specify the advanced options to achieve this goal. The following sections describe each of the options.

3.2.1 Emission mode

You can specify asynchronous (ASYNC) or synchronous (SYNC) to instruct CICS how events are emitted. This option does not apply to the Transaction start EP adapter.

With asynchronous event emission, the event is queued for asynchronous processing by an EP dispatcher thread, which is separate from the capturing transaction.

With synchronous event emission, event formatting and emission are completed within the unit of work of the capturing transaction. The EP adapter is part of the

application task. “Dispatch priority”, “Transaction ID”, and “User ID” are not applicable to the synchronous event emission mode.

Synchronous event emission is not supported for system events. CICS TS V4.2 or higher is needed if you specify synchronous emission mode.

3.2.2 Dispatch priority

You can specify Normal or High priority to control dispatching of events. High priority events are emitted when they are available. Normal priority events are emitted when they are available, but after any outstanding high priority events. This option is subject to the Events are Transactional setting.

3.2.3 Transaction ID

Specify the Transaction ID, which is not available for a custom EP adapter. The EP adapter program runs under this transaction ID. This is optional.

3.2.4 User ID

Specify a user ID so that the EP adapter transaction runs with this user ID. If you select the Use context User ID option, the EP adapter runs with the user ID under which the event was captured.

For more information about the effect on the EP adapter with different combinations of Transaction ID and User ID, see the CICS Information Center at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v5r1/index.jsp?topic=%2Fcom.ibm.cics.ts.eventprocessing.doc%2Ftasks%2Fdfhep_specify_dispatcher_information.html

When you install a BUNDLE resource that includes an event binding for which you specified a user ID in the Adapter tab of the CICS event binding editor, CICS checks that the user ID performing the install operation is authorized as a surrogate user of the user ID that is specified in the CICS event binding editor. This check also applies to the CICS region user ID during group list install on a CICS cold or initial start.

3.2.5 Transactional events

Specify whether events are transactional under the following conditions:

- Select the Events are transactional option if you want CICS to emit captured events only if the unit of work (UOW) that is associated with the event completes successfully. The events are emitted when there is an internal sync point within the program, or when the program ends and reaches an implicit sync point.

  If the transaction fails and backs out, the events are discarded.

- Do not select the Events are transactional option if you want CICS to process events that are associated with event binding in near-real time, regardless of whether the UOW commits.

You can assure the emission of an event by using an EP adapter with synchronous emission mode and the appropriate transaction mode. For example, WebSphere MQ EP adapter and TS queue EP adapter support synchronous and transactional event emission, while HTTP EP adapter and Transaction start EP adapter do not. For more information about the supported combinations of emission modes and transactional modes for EP adapters, see the CICS Information Center at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v5r1/topic/com.ibm.cics.ts.eventprocessing.doc/concepts/dfhep_event_processing_adapters.html#dfhep_event_processing_adapters__epadapteremissiontransmode

System events cannot be transactional events.

3.3 Predefined EP adapters

You can define an EP adapter separately from the event binding. By separating the definition of EP adapter from event binding, it is easier to change the configuration details for how events are to be formatted and emitted. The person defines EP adapters might be different from the person who defines event bindings. You can also reuse predefined EP adapters in multiple event bindings and EP adapter sets that were introduced in 3.4, “EP adapter set” on page 60.

Complete the following steps to create an EP adapter definition:

1. Right-click a CICS bundle project, select **New**, and the select **CICS Event Processing Adapter**, as shown in Figure 3-3 on page 58.

   The wizard that is used to create CICS EP Adapter configuration opens.

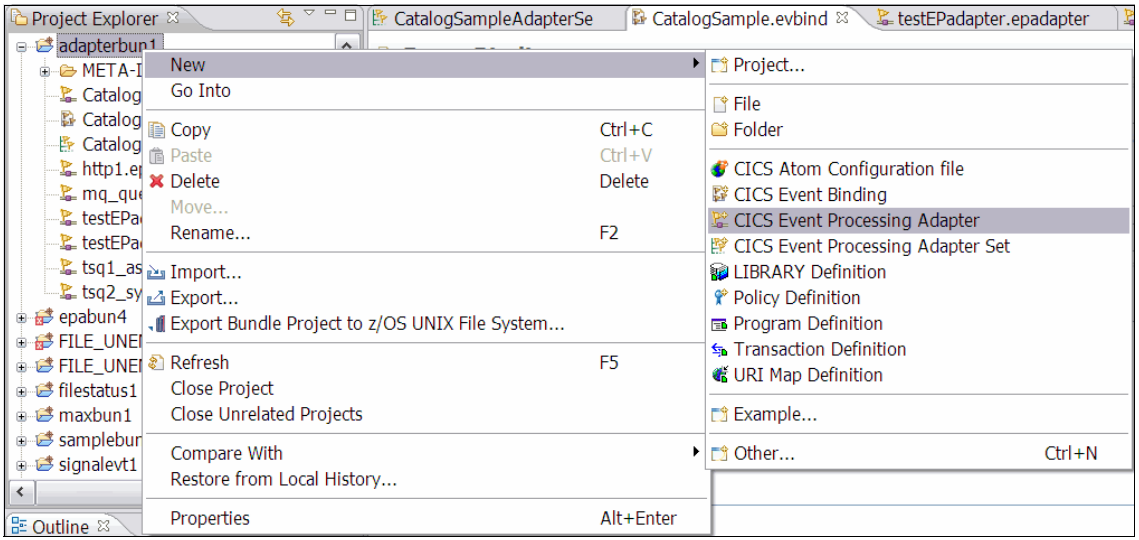


Figure 3-3 Creating an EP adapter

2. In the File name field, enter a name for the EP adapter and select **Finish**. The new EP adapter is opened for editing.

Separate EP adapters have the same options as when EP adapters are defined in an event binding. When the EP adapter definition is finished, you must install to CICS for it to emit events.

Figure 3-4 shows the event binding that uses a predefined EP adapter.

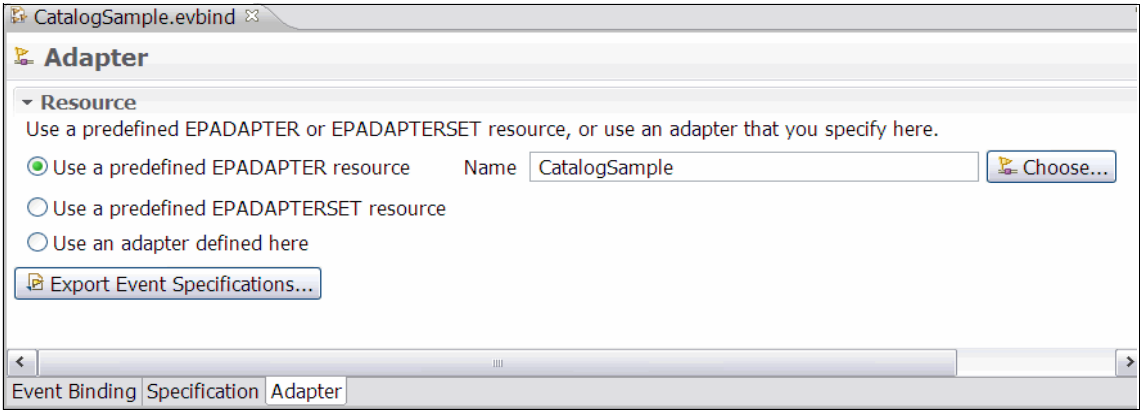


Figure 3-4 Event binding with separate EP adapter

Use either of the following methods to specify an EP adapter name:

- Enter an EP adapter name directly into the Name field of the Adapter tab.
- Click **Choose**, as shown in Figure 3-5. Use one of the following methods to specify the information:
  - Enter an EP adapter name.
  - Select an EP adapter from the workspace.

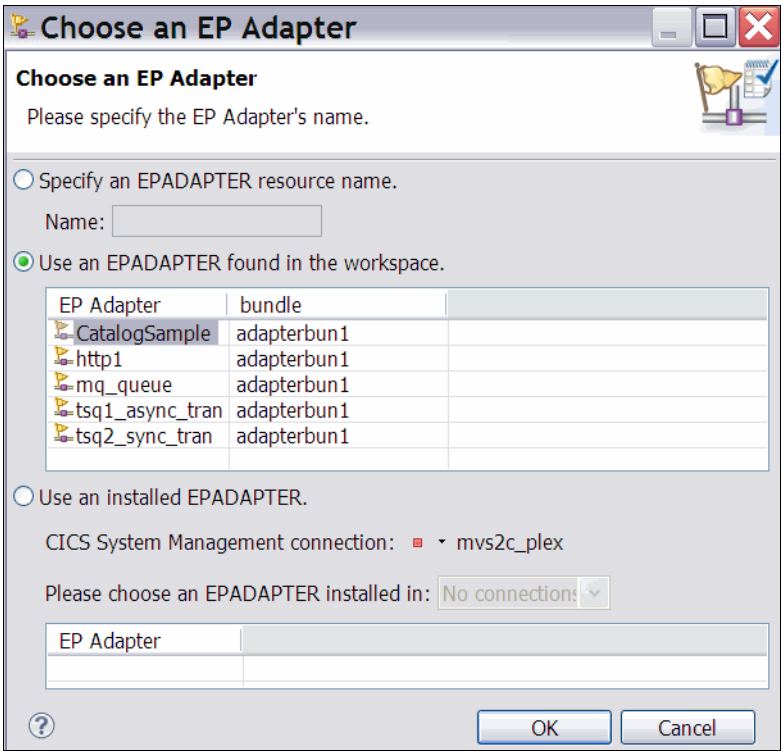


Figure 3-5 Choose an EP adapter in the workspace for event binding

  - If you have a CICS System Management connection, choose an installed EP adapter from the connected CICS, as shown in Figure 3-6 on page 60.

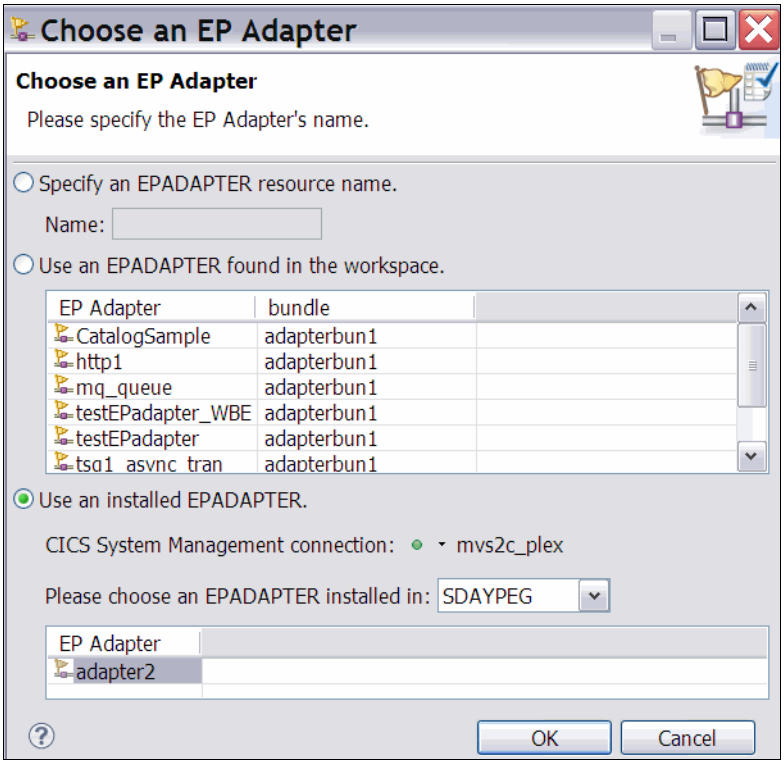


Figure 3-6 Choose an installed EP adapter for event binding

3.4 EP adapter set

If you want one event to be emitted to multiple users, use the EP adapter set as the adapter resource for the event binding. For example, you might want the events from the catalog sample event binding to be sent to a TS queue and an HTTP destination. In this case, you can define an EP adapter set that specifies the TS queue and HTTP EP adapters for the event binding instead of defining two event bindings with the same capture point.

An EP adapter set is an XML definition that contains one or more EP adapter names. By using an EP adapter set, you can format and emit events to multiple EP adapters. The definition and installation of EP adapter set is separate from those EP adapters whose names are contained in the EP adapter set. You must install and enable the EP adapter set and the EP adapters for the event to be emitted successfully to the EP adapters.

To create an EP adapter set definition, right-click a CICS bundle project as shown in Figure 3-3 on page 58 and choose **CICS Event Processing Adapter Set**, enter a name and then click **Finish**. The EP adapter set is opened for editing and you can add EP adapters to it.

Figure 3-7 shows an EP adapter set that contains two EP adapter names.

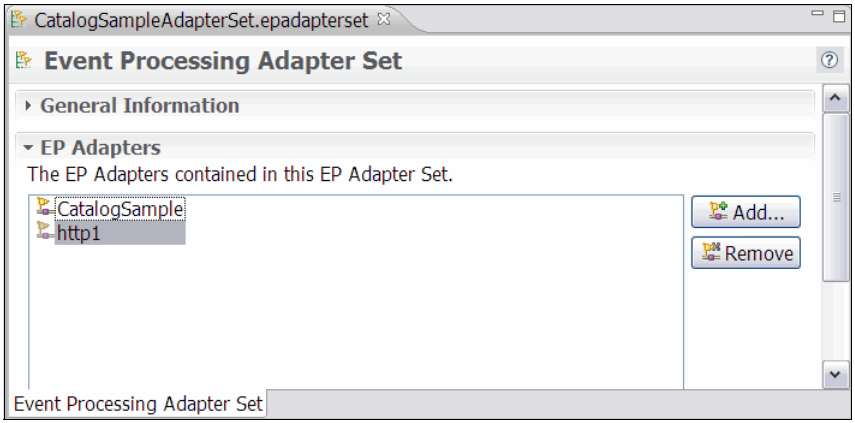


Figure 3-7 An EP adapter set definition

Specifying an EP adapter set in event binding is similar to specifying an EP adapter. Figure 3-8 on page 62 shows how to choose an EP adapter set in the CICS Explorer workspace.

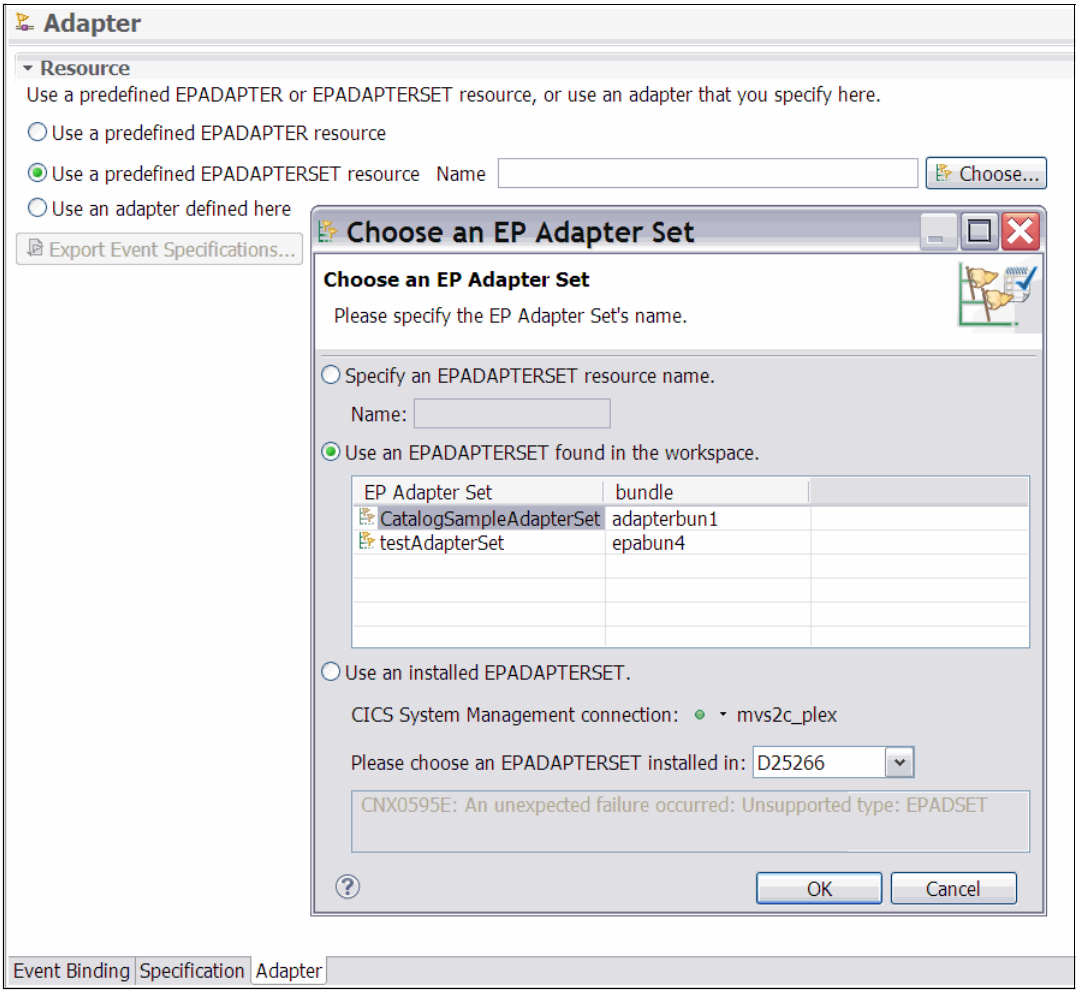


Figure 3-8 Choose an EP adapter set for event binding

3.5 Exporting event specifications

You can export descriptions for one or more event specifications in an event binding as a schema or copy book for use elsewhere.

If your chosen EP adapter or the adapter that is contained in EP adapter set emits events in a non-XML data format, such as CFE, the exported file type is a COBOL copy book (`.cpy`) file. The file name is the event specification name.

If your chosen adapter or the adapter that is contained in the chosen EP adapter set emits events in an XML data format (such as Common Base Event, Common Base Event REST, or WebSphere Business Events), the exported file type is an XML schema definition (`.xsd`) file. The file name is the event specification name that is appended with Common Base Event, Common Base Event REST, or WebSphere Business Events.

Complete the following steps to export event schema or copy book:

1. Click **Export Event Specifications**. The Export Event Specifications window opens.
2. Select the event specifications that you want to export.
3. Specify in the To Directory field a directory to which to export the event specifications.
4. If the event binding specifies an EP adapter but the EP adapter definition is not in the local workspace, the Data Format option is shown. Select the data format for export, as shown in Figure 3-9.

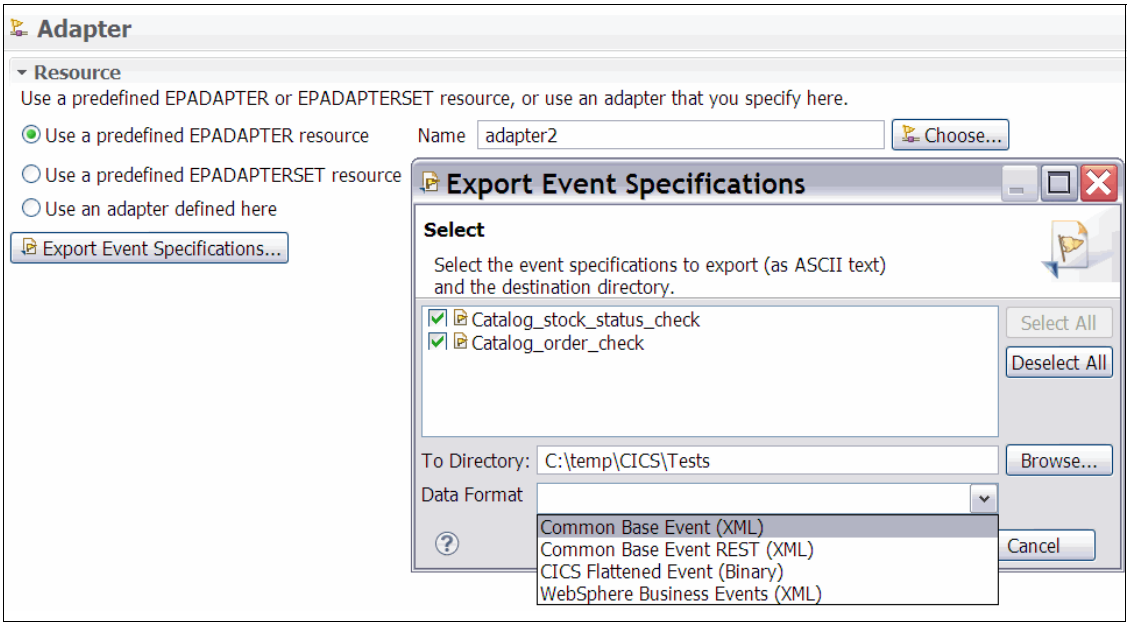


Figure 3-9 Select data format for export when the EP adapter does not exist in CICS Explorer workspace

5. If the event binding specifies an EP adapter set but one or more EP adapters that are specified by the EP adapter set are not in the local workspace, you can select the data formats, as shown in Figure 3-10 on page 64.

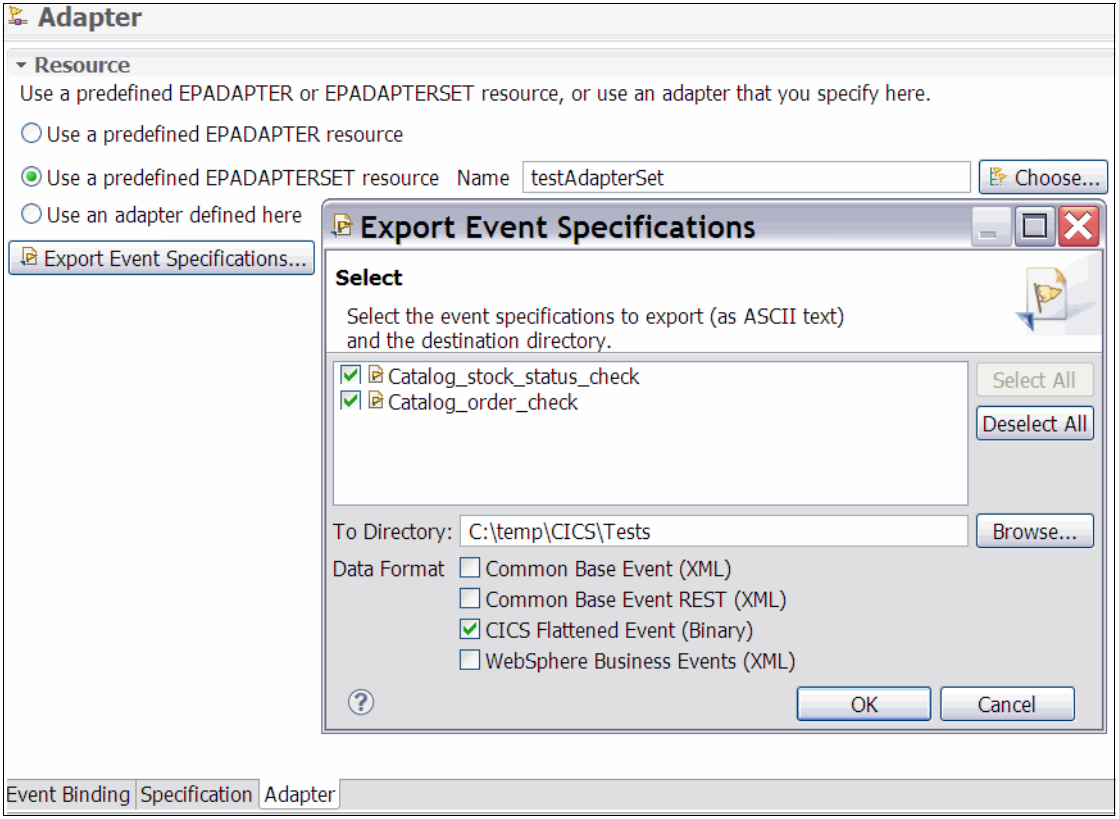


Figure 3-10 Selecting data format for export

6. Click **Export**.

One or more files are created in the specified directory for each event specification. If the event binding uses an EP adapter or uses an EP adapter set that contains only one EP adapter, only one file is created. If the event binding uses an EP adapter set that contains multiple EP adapters, one file is created for each of the data types of the EP adapters.

For example, if the EP adapter uses the Common Base Event format and you select two event specifications called `example1` and `example2` for export, two XML schema files are created, `example1_CBE.xsd` and `example2_CBE.xsd`. You can import the schema file to IBM Business Monitor to help define an inbound event.

If the EP adapter uses the WebSphere Business Events (XML) format, two XML schema files are created, `example1_WBE.xsd` and `example2_WBE.xsd`. You can use the schema file in the WebSphere Business Events Design Data tool to help define an event.

If the EP adapter uses CFE format, two COBOL copy books are created, `example1.cpy` and `example2.cpy`. You can use these copy books to process data in your event consumer programs.

For the event specification called `example1`, the files `example1_CBE.xsd`, `example1_WBE.xsd` and `example1.cpy` are created, if you specify an EP adapter set that has EP adapters with all of the data formats that were described in this chapter.

Part 2

Implementing CICS event processing

This part of the book focuses on the implementation of event processing in CICS. It includes a step-by-step guide to implementing CICS event processing with an overview of the environment that is used in our example. This part also includes a quick look at some troubleshooting ideas.

4

Environment overview

This chapter describes the environment that is used in our example in this book.

In addition, this chapter describes the sample application that is used to send events to IBM Operational Decision Manager and the IBM Business Monitor.

This chapter includes the following topics:

- Example environment
- Shopping sample application
- Sample scenarios
- The events emitted from the sample application

4.1 Example environment

To provide examples and demonstrate how to use event processing, a basic environment was created, as shown in Figure 4-1.

Tip: If you want to follow the examples that are shown, you must have access to an environment with similar components. However, the structure is flexible.

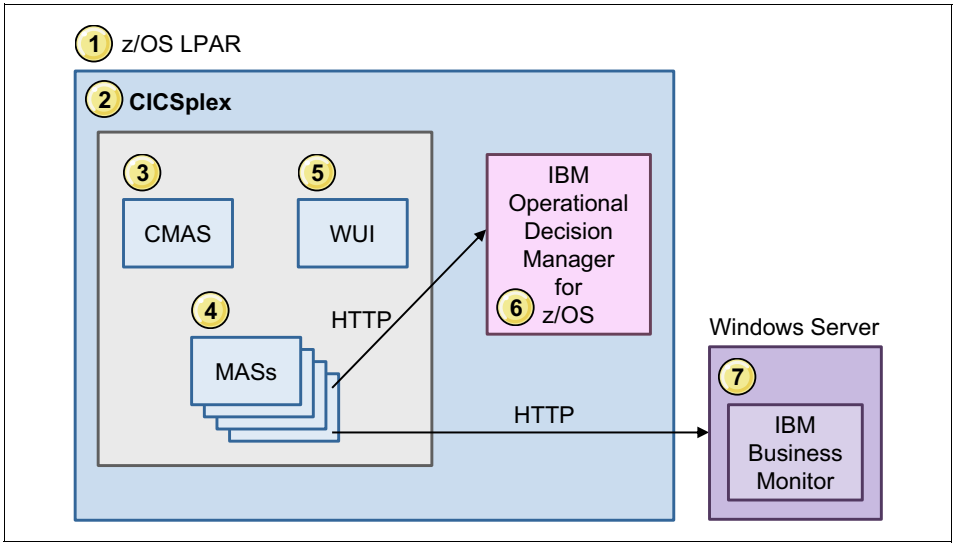


Figure 4-1 Shopping sample application environment

The following components are shown in Figure 4-1:

1. The example environment exists on a single z/OS logical partition (LPAR).
2. A single CICSplex enables CICS Explorer to connect through a web-based user interface and provide update facilities.
3. The CICSplex requires a maintenance point IBM CICSPlex® SM address space (CMAS).
4. The CICSplex has multiple CICS-managed address spaces; that is, CICS regions in which the example application runs and events are emitted
5. A web-based GUI is required to allow the CICS Explorer to connect to the CICSplex. To allow the update functions of CICS Explorer, the CMCIPORT(1491) parameter must be defined.

6. HTTP 1.1 compliant servers are required to allow events to be emitted from the CICS regions to both IBM Business Monitor (BM) and IBM Operational Decision Manager (ODM), by using HTTP EP adapters. The environment that is described in Figure 4-1 on page 70 has IBM ODM for z/OS that is running on the same z/OS LPAR as the CICS regions. However, in your environment, you can use any platform that is supported by IBM ODM.
7. The example environment has IBM Business Monitor installed on a Microsoft Windows Server.

4.2 Shopping sample application

To provide the examples, a simple order processing application was developed. This application is not intended to represent a real-life application. Instead, it is intended to show multiple possibilities for event processing API capture points. Figure 4-2 shows the order process that the example application implements.

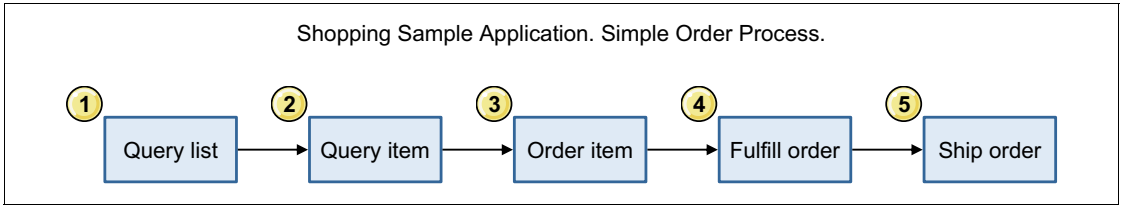


Figure 4-2 Shopping sample application order process

As shown Figure 4-2, the order process includes the following stages:

1. A customer displays a list of stock items on the panel.
2. A stock item is displayed that shows the stock level and price.
3. If satisfied with the selection, the stock item is ordered and the order is posted to the ORDER file.
4. The warehouse takes orders and fulfills them by reducing the stock levels.
5. The warehouse marks the order as shipped.

A set of programs, transactions, BMS maps, and copy books were developed to implement this order process. The structure of the application is shown in Figure 4-3 on page 72,

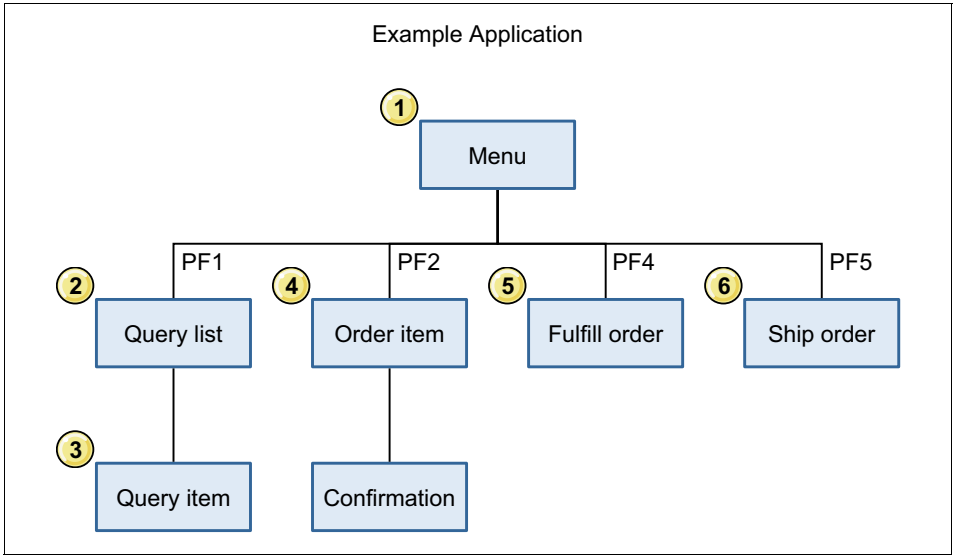


Figure 4-3 Shopping sample application structure

The following components of the Shopping sample application are shown in Figure 4-3:

1. Transaction and program MENU provides the ability to drive the various functions of the application. The BMS map (MENU1) has a field to enter a five-digit customer number, which is required for the query list and order item functions.
2. From the menu, providing a customer number and pressing PF1 starts the psuedo-conversational transaction QRY and program QUERY. This initially brings up a list of stock items from the STOCK file.
3. By using any character to select a stock item from the list, displays the stock level and price of the item.
4. From the menu, providing a customer number and pressing PF2 starts the conversational program ORDER. This allows the users to enter the stock item ID and quantity to order. Program SENDORDR is called to generate a record on the ORDER file and a confirmation panel opens.
5. From the menu, pressing PF4 simulates the warehouse fulfilling all outstanding orders by running program fulfill. A message is displayed on the menu detailing how many items are fulfilled and the total value of all the orders.
6. From the menu, pressing PF5 simulates the warehouse shipping all fulfilled orders by running program SHIP. A message is displayed on the menu detailing how many items were shipped.

The shopping sample application with all its source and setup job control language (JCL) can be downloaded from the Redbooks website. The provided `readme.txt` file provides details about how to upload the files to z/OS and configure and run the setup JCL.

4.3 Sample scenarios

To allow the next chapters to provide semi-realistic demonstrations, the following four scenarios were developed to be used with the example application:

- Query versus sale
- Stock low
- Shipped order meets service level agreement (SLA)
- High-value order breakdown

4.3.1 Scenario: Query versus sale

This scenario emails a discount offer to a customer if the customer queried a stock item repeatedly (four times), but did not order it. The theory behind this idea is that the customer is interested in an item but might need some encouragement to complete the sale. Figure 4-4 shows the flow of this scenario.

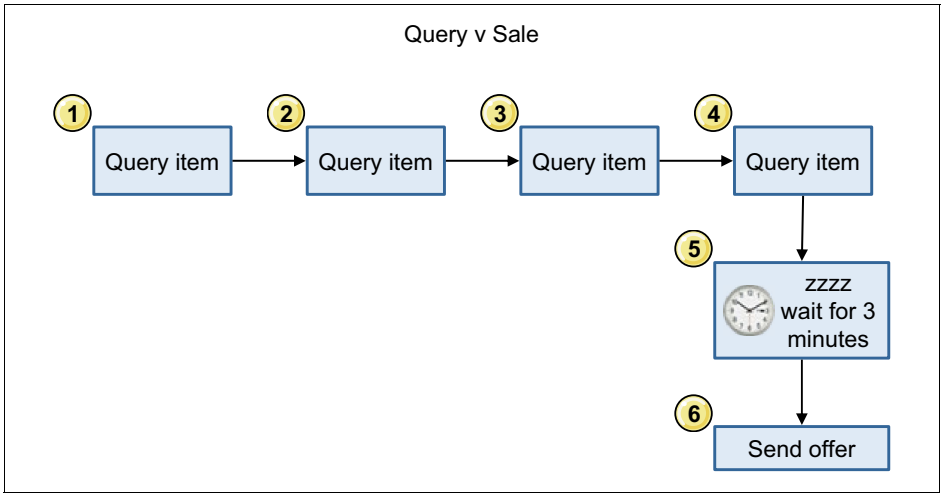


Figure 4-4 Query versus sale scenario

The scenario that is shown in Figure 4-4 on page 73 includes the following stages:

1. The customer queries the item for the first time.
2. The customer repeats the query.
3. The customer repeats again.
4. The customer repeats the query for the fourth time.
5. The scenario waits for a period (in the examples, this period is three minutes for demonstration purposes).
6. If an order with the same stock ID and customer number was been received in that time, an email is sent with a discount offer.

For this scenario to work, two types of events must be emitted from the example application: four of the query events, each emitted whenever a query item is performed, and one of the order events when an order is placed.

This type of scenario is best-suited for IBM ODM to process the events that are emitted from CICS TS. The HTTP EP adapter is used to send events from CICS TS to the ODM server.

4.3.2 Scenario: Stock low

This scenario sends an event whenever an order is fulfilled and the stock level is below 50 units. This event alerts the warehouse through a dashboard that the item must be restocked. Figure 4-5 shows the flow of the scenario.

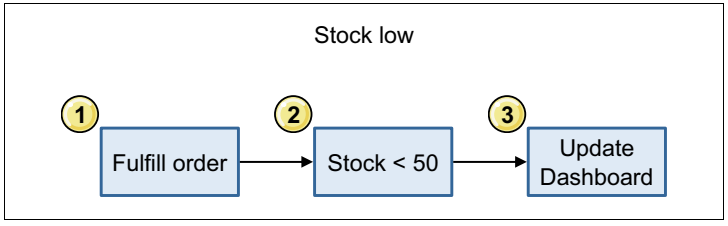


Figure 4-5 Stock flow scenario

The scenario that is shown in Figure 4-5 includes the following stages:

1. The fulfill process allocates stock to each outstanding order.
2. The stock level is under 50.
3. Update a dashboard with a request to replenish the stock.

For this scenario to work, an event must be emitted whenever the stock record is updated and the level is below 50.

This type of scenario is best-suited for IBM Business Monitor to process the events emitted from CICS TS and will also use the HTTP EP Adapter.

4.3.3 Scenario: Shipped order meets service level agreements

This scenario can be used to monitor the warehouse's efficiency. Figure 4-6 shows the flow of the scenario.

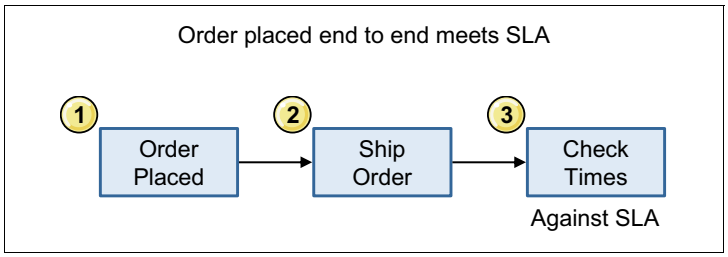


Figure 4-6 Order end to end scenario

The scenario that is shown in Figure 4-6 includes the following stages:

1. An order is placed.
2. The warehouse ships the order.
3. The times are compared and a Key Performance Indicator is updated to reflect the warehouse's efficiency.

Events are to be emitted whenever an order is placed and shipped. The order placed event that is used in 4.3.1, "Scenario: Query versus sale" on page 73 can be reused here.

This type of scenario is best-suited for IBM Business Monitor to process the events emitted from CICS TS and use the HTTP EP Adapter by using the Common Base Event format.

4.3.4 Scenario: High-value order breakdown

This scenario allows a dashboard to present a breakdown of what high-value orders are being placed within the system. The dashboard can be configured to present an order distribution that is based on demographics, customer importance, order values, and hot stock items. Figure 4-7 shows the flow of the scenario.

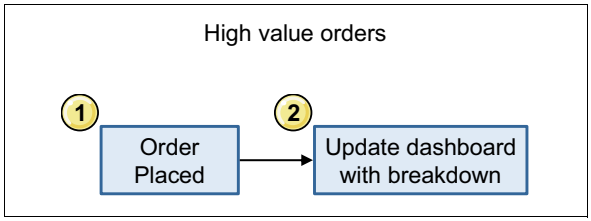


Figure 4-7 High value order scenario

The scenario that is shown in Figure 4-7 includes the following stages:

1. An order is placed.
2. The dashboards are updated with the new breakdowns.

An event is emitted whenever an order is placed. Because the items of data that are required to be emitted with the event are more complex than those that are available at a standard capture point in this application, this scenario must be an intrusive change to the application code so that all the data can be collated and emitted together.

This type of scenario is best-suited for IBM Business Monitor to process the events emitted from CICS TS and will use the HTTP EP Adapter.

4.4 The events emitted from the sample application

The capture points in the examples are based on the CICS API calls made in each phase of the order process in the shopping sample application (Query stock, Order Item, fulfill Order, and Ship Order). Table 4-1 on page 77 shows the information that is contained in the capture specifications for the sample event binding. The Emitted Business Information column contains the payload data that is captured with the events, which is selected from the containers that are passed with each API command.

Table 4-1 The capture points in the sample MENU application

Event Name	Capture Point	Predicates	Emitted Business Information
QueryStock	Program Initiation	Program Name = QUERY	CustomerNumber: ► Numeric Packed-Decimal ► Capture Length = 3 ► Format Length = 6 StockId: ► Numeric Packed-Decimal ► Capture Length = 3 ► Format Length = 6
Order	LINK Capture after command.	Program Name = SENDORDR	CustomerNumber: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6 OrderNumber: ► Numeric Packed-Decimal ► Capture Length = 3 ► Format Length = 6 StockId: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6 Quantity: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6
fulfill	PUT CONTAINER	Program name = UPDSTOCK Container name = OUTPUT Stock item level is < 50	StockId: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6 OldLevel: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6 NewLevel: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6

Event Name	Capture Point	Predicates	Emitted Business Information
Ship	REWRITE	Program name = SHIP Response Code = Ok File name = ORDER Order status = ‘S’	OrderNumber: ► Numeric Packed-Decimal ► Capture Length = 3 ► Format Length = 6 CustomerNumber: ► Numeric Packed-Decimal ► Capture Length = 3 ► Format Length = 6 StockId: ► Numeric packed-Decimal ► Capture Length = 3 ► Format Length = 6 Quantity: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format length = 6

The capture points are based on the API calls in the shopping sample application programs, as shown in Table 4-2.

Table 4-2 API commands in the MENU sample application, on which capture points are based

Event Name	Capture Point	MENU Sample program Name	API command
QueryStock	PROGRAM INIT	QUERY	EXEC CICS RETURN TRANSID(‘QRY’) CHANNEL(‘LIST-STOCK’) END-EXEC.
Order	LINK Capture after command.	SENDORDR	EXEC CICS LINK PROGRAM(‘SENDORDR’) CHANNEL(CHANNEL-NAME) END-EXEC.
Fulfill	PUT CONTAINER	UPDSTOCK	EXEC CICS PUT CONTAINER(‘OUTPUT’) FROM(UPDATE-OUT) END-EXEC.
Ship	REWRITE	SHIP	EXEC CICS REWRITE FILE(‘ORDER’) FROM(ORDER-REC) END-EXEC.

In addition to the above non-invasive capture points that are shown in Table 4-2 on page 78, we included the EXEC CICS SIGNAL EVENT API call that is shown in Example 4-1 in the SENDORDR sample program to demonstrate how this technique of capturing events might be used if you can alter your source code to capture events.

Example 4-1 EXEC CICS SIGNAL EVENT API call

```
EXEC CICS SIGNAL EVENT('SendOrder')
     FROM(SIGNAL-EVENT-DATA)
     FROMLENGTH(LENGTH OF SIGNAL-EVENT-DATA)
     END-EXEC.
```

The SIGNAL EVENT details are shown in Table 4-3.

Table 4-3 SIGNAL EVENT capture point in Shopping sample application

Event Name	Capture Point	Predicate	Emitted Business Information
SendOrder	SIGNAL EVENT	Event name = SendOrder	CustomerNumber: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6 OrderNumber: ► Numeric Zoned-Decimal ► Capture Length = 5 ► Format Length = 6 CustomerName: ► Text Character ► Capture Length = 50 ► Format Length = 50 City: ► Text Character ► Capture Length = 50 ► Format Length = 50 Country: ► Text Character ► Capture Length = 50 ► Format Length = 50 Premium: ► Text Character ► Capture Length = 1 ► Format Length = 1 TotalOrderValue: ► Numeric Zoned-Decimal ► Capture Length = 11 ► Format Length = 13 ► Precision = 2

5

Setting up CICS for events

This chapter describes how to set up CICS for events. It shows how to configure CICS TS for event processing and use the CICS Explorer to control event processing. The CICS Explorer provides the development tooling environment that is needed to create an event binding.

This chapter assumes that you already have a CICS bundle deployed on zFS that contains an event binding. You are shown how to deploy, install, and manage bundles in CICS. For more information about how to create bundles and event bindings, see Chapter 6, “Capturing application events” on page 107.

This chapter also provides information about the various CICS security options that relate to CICS event processing. An example of how to protect access to bundles and event bindings also is provided. The External Security Manager (ESM) that is used in this case is IBM Resource Access Control Facility (RACF®).

This chapter includes the following topics:

- CICS Explorer setup
- CICS System setup
- Creating and installing a bundle definition
- Enabling and disabling and discarding events
- Security considerations for CICS events

5.1 CICS Explorer setup

This section describes how to obtain the CICS Explorer and connect it by using the CICS management client interface (CMCI) to the IBM CICSPlex SM web User Interface (WUI).

For more information about the use of and connecting the CICS Explorer to CICS, see the CICS Information Center at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v5r1/topic/com.ibm.cics.ts.installation.doc/topics/explorer_configure_connection.html

5.1.1 Obtaining the CICS Explorer

This section shows you how to install and configure CICS for event processing by using the CICS Explorer. This chapter uses CICS Explorer V5.1, which can be used to manage CICS TS V5.1 and V4. The CICS Explorer can be downloaded from this website:

http://www.ibm.com/software/htp/cics/explorer/

After you install the CICS Explorer, use the EPREDW section to connect it to our CICSPlex SM WUI.

5.1.2 CICS Explorer connectivity

Complete the following steps to connect to our CICSPlex SM WUI server:

1. In the CICS Explorer Host Connections view, highlight **CICS System Management** and click **Add**. Select **CMCI** from the menu, as shown in Figure 5-1.

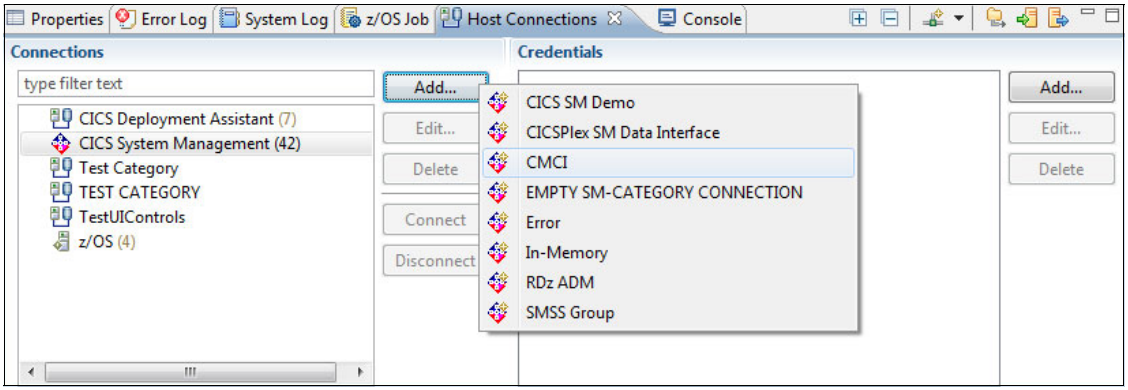


Figure 5-1 CICS Explorer Host Connections view: Adding a CMCI connection

Important: We use the connection type `CICS Management Interface` to enable update capability. The default connection type `CICSPlex SM Data Interface` allows read-only capability.

Our WUI server (Hostname: `wtsc66.itso.ibm.com`) has port `1491`. The `CMCIPORT` system initialization parameter for our WUI is set as `CMCIPORT=1491`, as shown in Figure 5-2 on page 84.

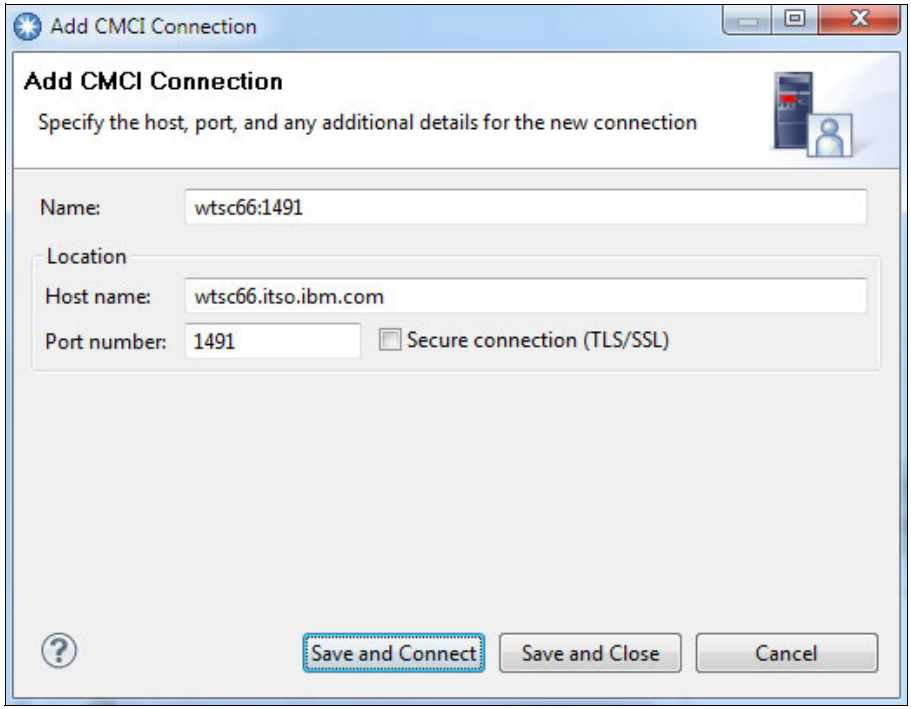


Figure 5-2 Add CMCI Connection dialog

For more information about how to set up the CMCI in CICS, see this website:

`http://pic.dhe.ibm.com/infocenter/cicsts/v5r1/topic/com.ibm.cics.ts.clientapi.doc/topics/clientapi_setup.html`

2. Click **Save and Connect**. You are prompted to enter your user name and password, if required.

The CICS Explorer is now connected to your CICSPlex via a CMCI Connection.

5.1.3 The CICS event binding editor

The CICS event binding editor is supplied as part of the CICS Explorer, which was downloaded in 5.1.1, “Obtaining the CICS Explorer” on page 82. Chapter 6, “Capturing application events” on page 107shows how this is used to create event bindings.

5.2 CICS System setup

This section reviews the CICS system setup that is required to implement events in CICS.

5.2.1 Adding TCP/IP support to use the HTTP EP adapter

The shopping application uses the HTTP EP adapter to format and process events. To use the HTTP EP adapter, CICS must be enabled for TCP/IP.

In the CICS SIT, set `TCPIP=YES`, as shown in Example 5-1.

Example 5-1 CICS SIT showing TCP/IP

```
APPLID=EPREDA01
SYSIDNT=RED7
START=INITIAL
GRPLIST=(DFHLIST,EPREDA01)
TCPIP=YES
```

5.2.2 Enabling CICS event processing

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

When a `START=WARM` or `START=EMERGENCY` parameter is used, the settings from the previous run of CICS are used. So, unless the setting changed, event processing should be enabled. You can inquire on EVENTPROCESS to check this; for example, by using the CICS Explorer.

You can stop and start event processing from the IBM CICS Explorer, the CICSPlex SM WUI, or the CICS SPI or API commands. For example, you might want to stop event processing during an upgrade or system maintenance and then restart event processing.

CICS event processing can be set by using three possible states: `START`, `DRAIN`, or `STOP`.

DRAIN causes event capture to stop but allows events that were captured to be processed through the system and emitted. Transactional events are not emitted if the unit of work had not reached sync point at the time the DRAIN request is received. STOPPED causes event processing to stop immediately, with no further events captured or emitted.

5.2.3 Stopping CICS event processing

Complete the following steps to stop event processing in the CICS region:

1. By using the **CICSplex Explorer** view, select the CICSplex or CICS region for which you want to stop event processing.
2. By using the IBM CICS Explorer toolbar, click **Operations** → **Event Processing** → **Event Processing** to open the **Event Processing** view, as shown in Figure 5-3. The current status of event processing is shown.

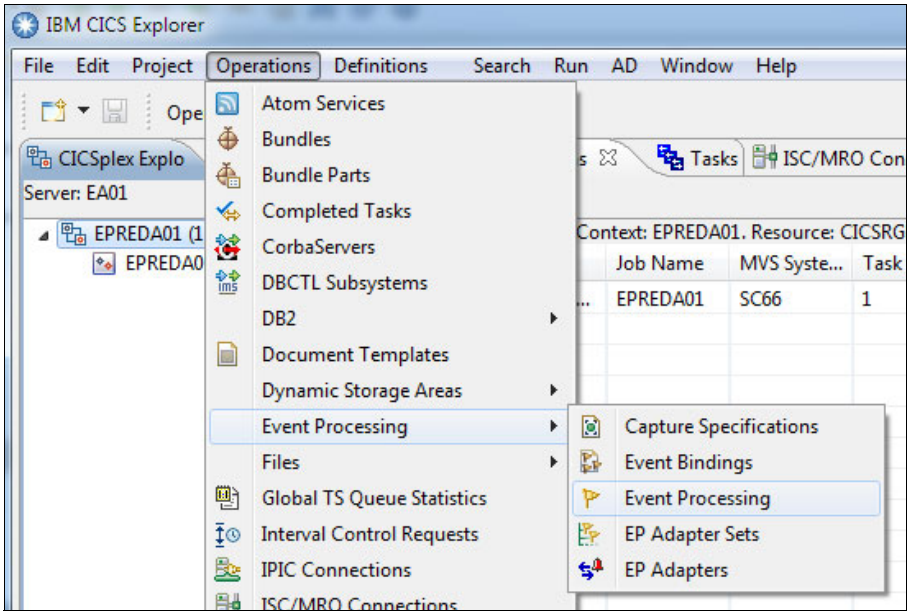


Figure 5-3 Opening the Event Processing view in the CICS Explorer

3. Right-click the region where you want to stop event processing and click **Stop**, as shown in Figure 5-4 on page 87.

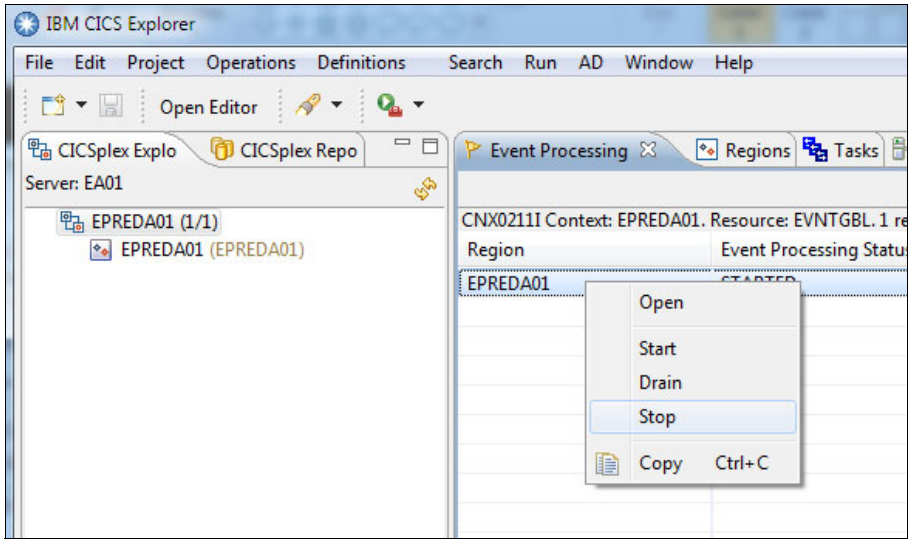


Figure 5-4 Stopping Event Processing in the CICS Explorer

4. A window opens in which you can confirm the action that you are about to perform, as shown in Figure 5-5. Click **OK** to continue.

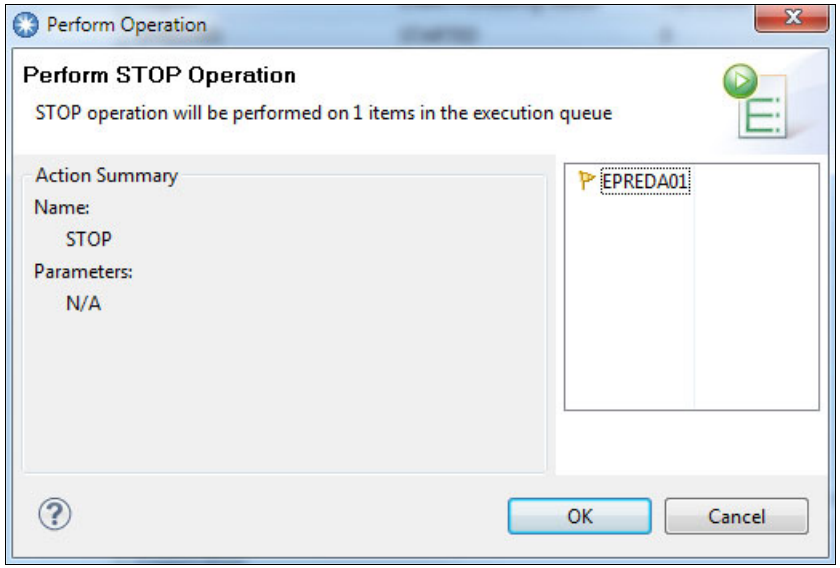


Figure 5-5 Perform action confirmation

The STOPPED status displays, which indicates that event processing is stopped, as shown in Figure 5-6.

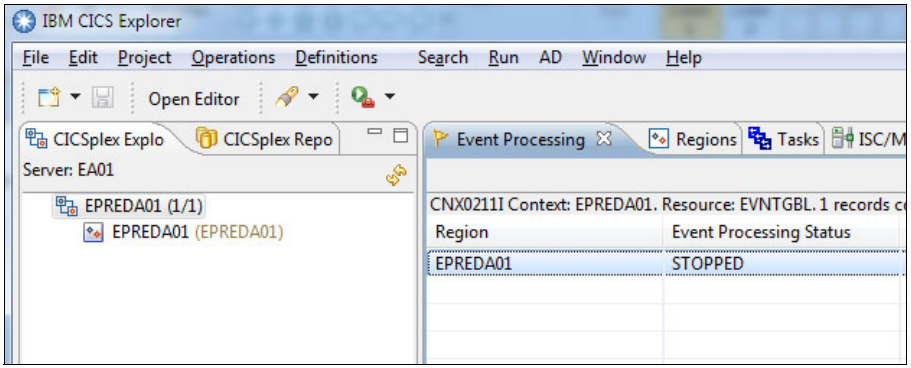


Figure 5-6 Event Processing Stopped

5. To restart event processing, right-click the region where you want to start event processing and click **Start**. Click **OK**.

5.3 Creating and installing a bundle definition

This IBM Redbooks publication uses the CICS Explorer to define and install bundle resources. Other mechanisms (such as CEDA) also can be used.

The following tasks are performed by using CICS Explorer:

- Create a bundle resource definition (BUNDDEF resource).
- Install a bundle resource into CICS.

Bundles and event bindings

A bundle resource is a type of resource that was introduced in CICS TS V4.1. A bundle resource defines a bundle, which is a unit of deployment for an application. A bundle is a collection of CICS resources, artifacts, references, and a manifest that you can deploy into a CICS region to represent an application. CICS bundles can describe various CICS resources types, one of which is an event binding.

A bundle is deployed to z/OS UNIX file system (or zFS) and is a directory structure that contains artifacts. The bundle resource defines where the bundle is deployed on z/OS UNIX and its status.

To deploy your event binding file to a CICS system, you must first export the bundle that contains it to zFS. You then install the bundle resource that points to the bundle on the zFS. This can be done by using IBM CICS Explorer, the CICSPlex WUI, the RDO CEDA transaction, or the DFHCSDUP utility.

5.3.1 Creating a new CICS bundle definition

This section describes how to define a bundle resource that is named `SHOPEVE`, which references the sample bundle that is called `ShoppingEventBundle_1.0.0`, as described in Appendix B, “Additional material” on page 243. You should extract the bundle to an zFS directory, as shown in Example 5-2.

Example 5-2 Sample event binding in sample bundle

```
/u/cicsrs2/bundles/ShoppingEventBundle_1.0.0/
  Type  Filename
_ Dir   .
_ Dir   ..
_ Dir   META-INF
_ File  ShoppingEventBinding.evbind
```

Creating and deploying bundles: When you create and deploy your bundles to zFS by using the CICS Explorer, you should ensure proper management of the bundle source code. The bundle cannot be reconstructed from the exported data in zFS, and a failure of your workstation can cause the data to be lost. You can use the Export function of CICS Explorer to export the bundle and check it in to a source code management system.

Use the New Wizard in CICS Explorer to create a bundle resource definition that can then be installed in one or more active CICS systems, as shown in Figure 5-7 on page 90. For the purposes of this example, you should use CICS TS V4.2 or higher. This is because the sample bundle contains features (such as a separate EP adapter) that were introduced in V4.2.

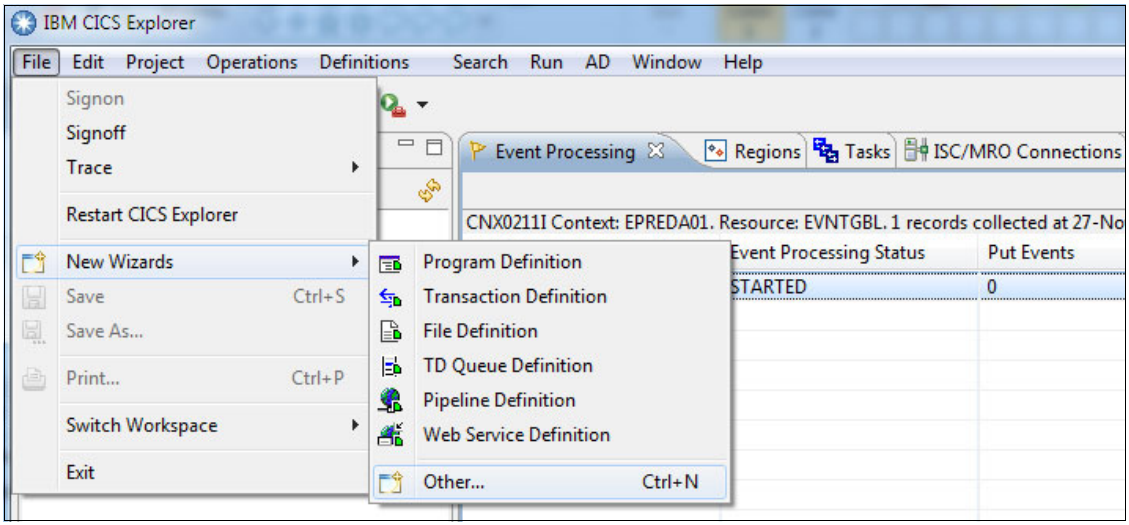


Figure 5-7 Opening New Wizard

You enter the basic details that are needed to create the resource by using the New Wizard and add information by using the editor. You must be connected to a CICS version V4.1 (or above) system to create resource definitions.

Complete the following steps to create a resource:

1. Open the New Wizard by completing the following steps:
   a. Click **Explorer** → **New Wizards** → **Other**.
   b. Select **Bundle Definition** and click **Next**, as shown in Figure 5-8.

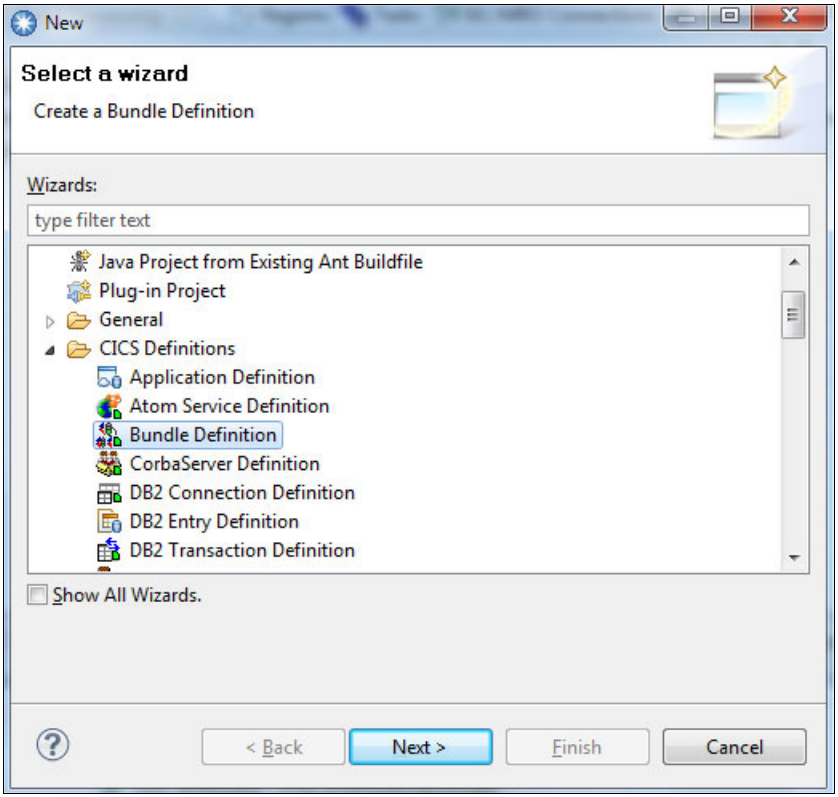


Figure 5-8 Creating a BUNDDEF

2. The New Bundle Definition window opens, as shown in Figure 5-9.

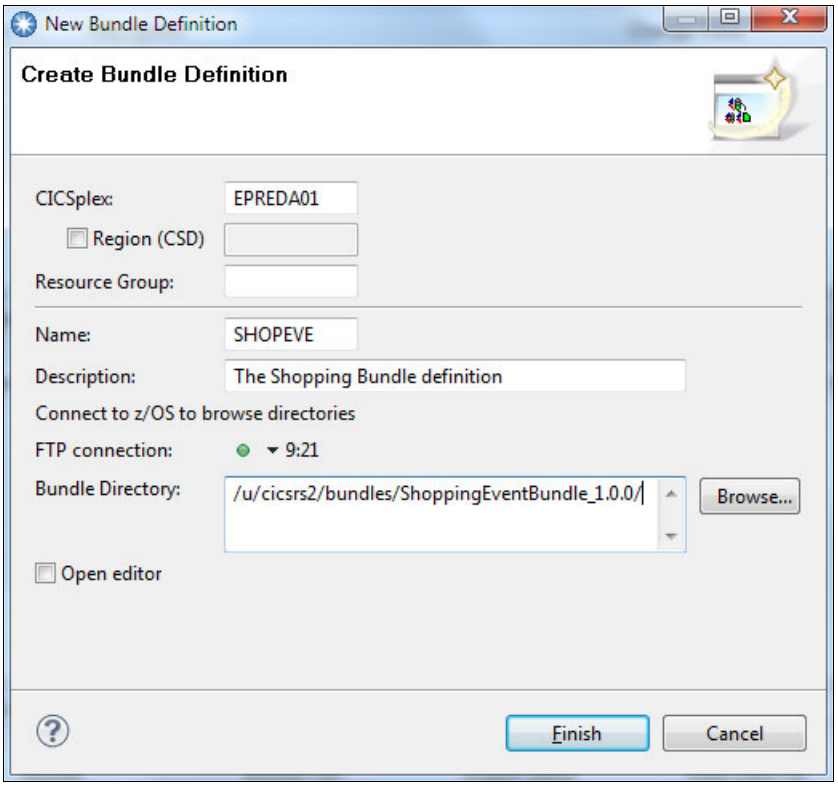


Figure 5-9 Specifying bundle definition options

3. If you selected a CICSplex before you opened the wizard, the CICSplex field contains the name of the CICSplex. You can type over the name or, if the field is empty, enter a new name. The light bulb symbol next to the text field shows that content assist is available. When you press Ctrl+space, content assist displays a list of the possible choices for the field. You can double-click a name in the list to select the CICSplex.
4. If you selected a resource group before you opened the wizard, the Resource Group field contains the name of the resource group, and the CICSplex field contains the name of the CICSplex. You can type over the name or, if the field is empty, enter a new name. In this case, the Resource Group field is left blank so as not to create a resource group.

5. Complete the remaining fields in the wizard. This example specifies a Bundle Directory value of `/u/cicsrs2/bundles/ShoppingEventBundle_1.0.0/`, because that was the zFS directory where the bundle was previously exported.
6. If you want to save the new resource and immediately open it in the editor, ensure that the Open editor option is selected.
7. Click **Finish** and your bundle definition is created. In this case, the new definition is saved onto the CPSM repository.

> **New definition:** When you are creating a definition with the CICS Explorer, the definition panel that opens for your specified new resource prompts you to supply the required parameters to complete the definition. Field help is available by pressing the PF1 key on any of the parameter fields. If you want to define further options for your definition, select the open editor and click **Finish**. The editor opens and you can specify other attributes for your definition

5.3.2 Installing a bundle definition into CICS

When a bundle is installed into CICS, the various resources that it contains are each installed as a result. This allows the resources in the bundle to be managed as a group. To install an event binding into CICS, install the bundle in which it is included.

When an event binding is installed, the capture specifications that are associated with the events in the binding are deployed into CICS and the run time starts to capture events where they match the filtering that is specified in the capture specifications.

An event binding is installed by installing the bundle resource for the bundle that contains it. If the bundle resource specifies `enabled` as its initial state, this also enables the event binding if the install action succeeds.

Complete the following steps to install the bundle by using the IBM CICS Explorer:

1. From the IBM CICS Explorer toolbar, click **Definitions** → **Bundle Definitions** to view the list of bundle resource definitions, as shown in Figure 5-10 on page 94.

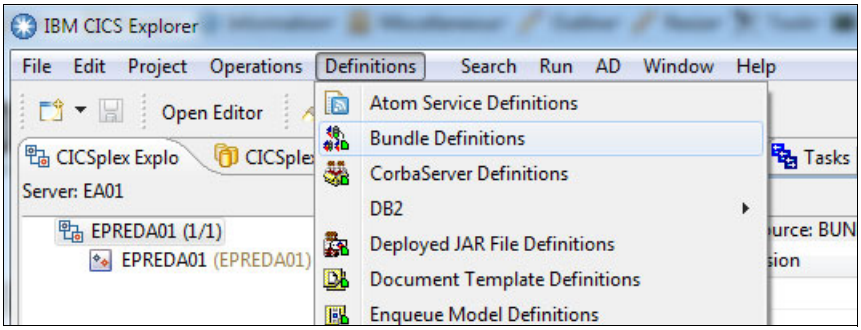


Figure 5-10 Bundle definitions to view the list of BUNDDEF resource definitions

2. Right-click the event binding bundle definition name in the Bundle Definitions view and click **Install**, as shown in Figure 5-11. On the Install action panel, select the appropriate target CICS system or system group option where you want to install your event binding bundle and click **OK**.

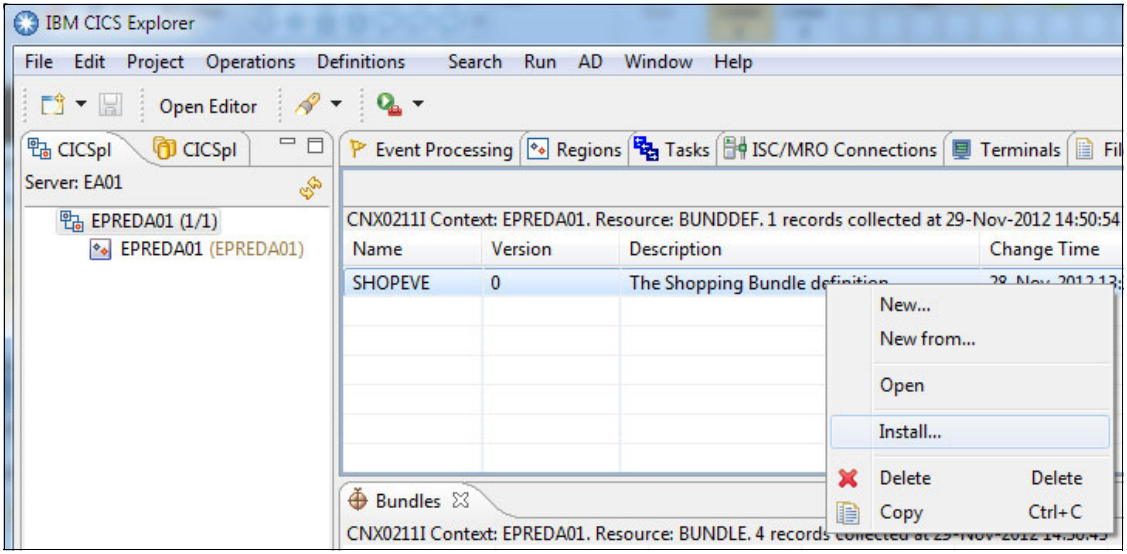


Figure 5-11 Install bundle definitions

The message `CNX0551I Install of BUNDDEF definitions into EPREDA01 successful` displays. The bundle that contains the event binding installs in the specified CICS region. CICS also dynamically creates any other resources that are defined in the bundle.

What to do next

After the resource definition installs successfully, you can view the status of installed event bindings or bundles. To do this, open the various resource views from the Operations menu on the CICS Explorer toolbar. Figure 5-12 shows the various views that are available.

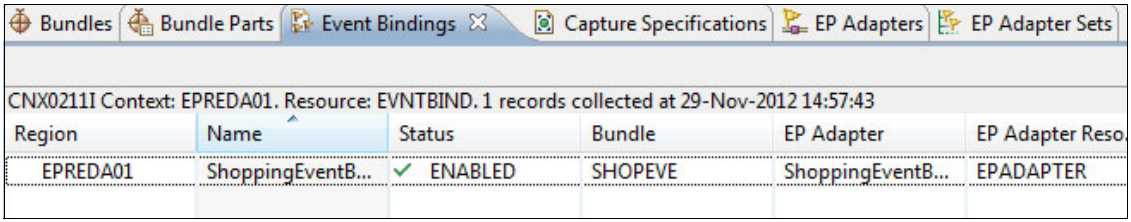


Figure 5-12 Operational views for Bundles and Event Processing resources

5.4 Enabling and disabling and discarding events

This section shows how to enable, disable, and discard event bindings or bundles by using the CICS Explorer.

If you disable an event binding resource, CICS also disables the bundle resource that contains it. However, any other resources that are part of the bundle remain in an enabled state in the CICS region. If you re-enable the event binding successfully, the bundle resource also changes to the enabled state.

Tip: Because a bundle is the unit of deployment for an application, it is recommended that you enable, disable, and discard the bundle resource only. However, it can still be useful to disable individual event bindings to effectively turn off the emission of particular events.

Complete the following steps to disable and discard an event binding:

1. Select **Operations** → **Event Processing** → **Event bindings** or **Operations** → **Bundles** to open the Event Bindings or Bundles views, as shown in Figure 5-13 on page 96.

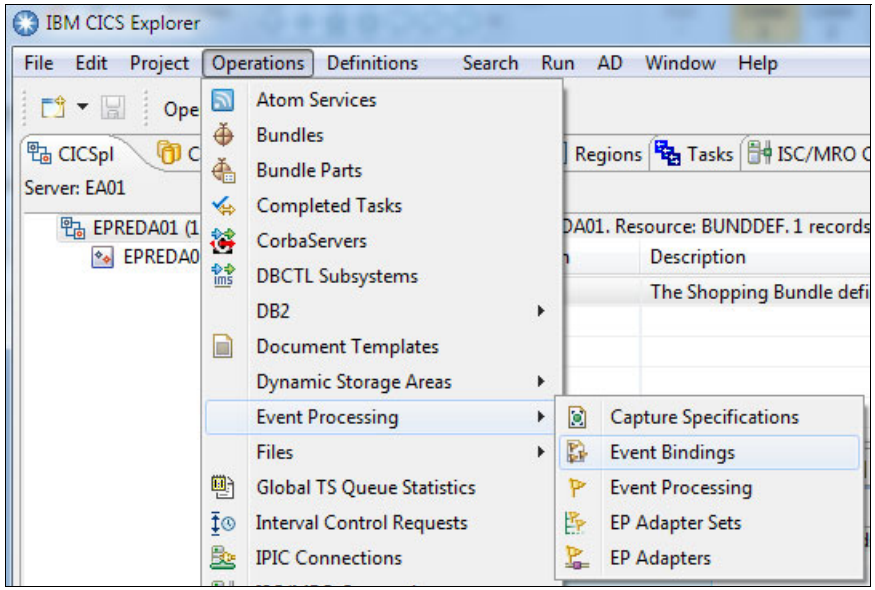


Figure 5-13 Working with event bindings

2. You can disable event bindings or bundles right-click in these views. Click to select your event binding, then right-click to disable, as shown in Figure 5-14.

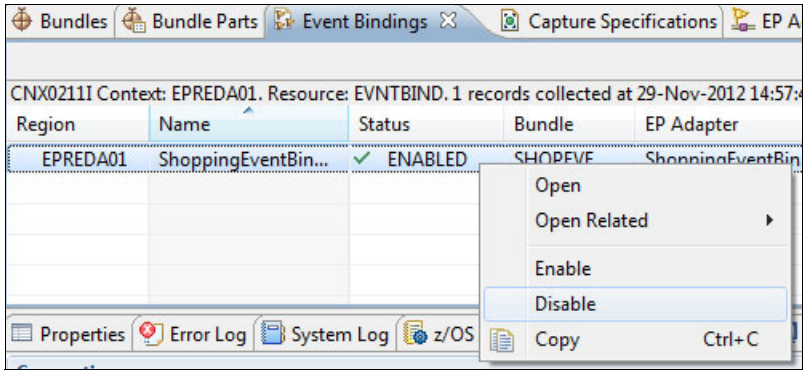


Figure 5-14 Disabling the ShoppingEventBinding event binding

3. Click **OK**, as shown in Figure 5-15.

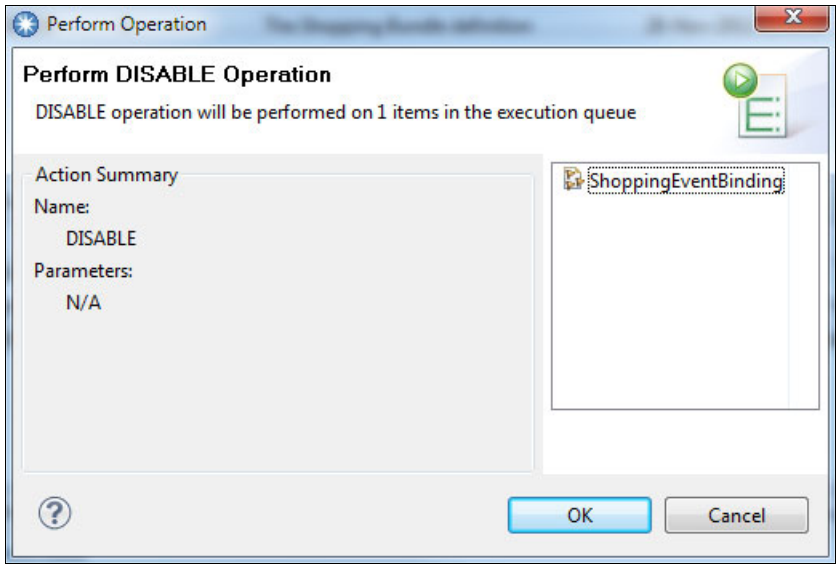


Figure 5-15 Confirmation panel for disable

The event binding is disabled, as shown in Figure 5-16.

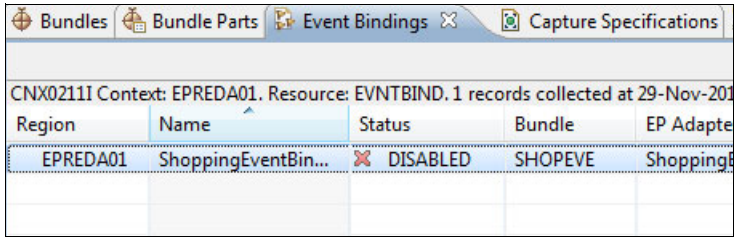


Figure 5-16 Disabled ShoppingEventBinding event binding

4. In the Bundles view, you can now right-click and discard SHOPEVE bundle, as shown in Figure 5-17 on page 98.

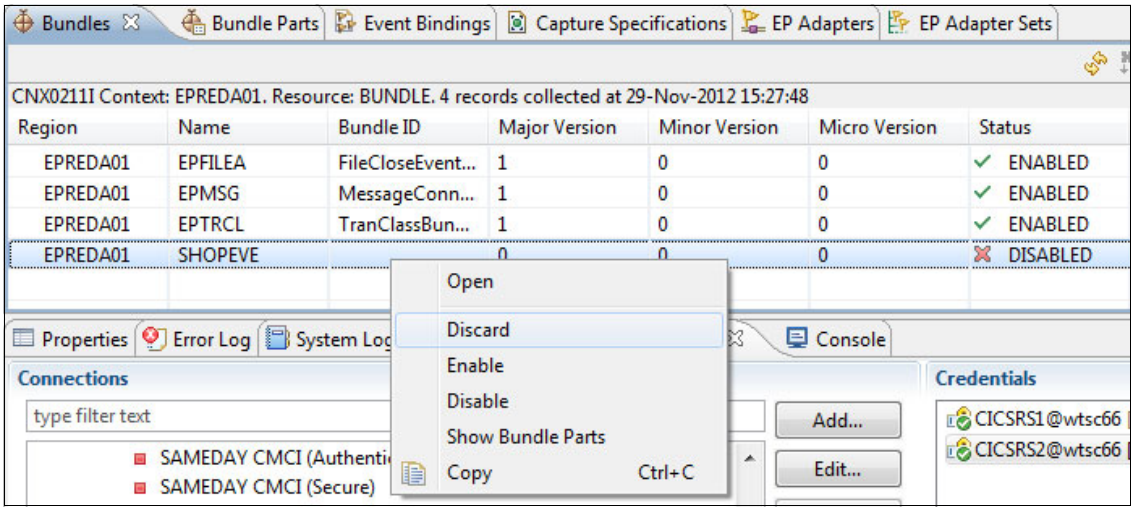


Figure 5-17 Discarding a disabled bundle

5. Click **OK** and the bundle resource is discarded, as shown in Figure 5-18. The event binding also is discarded.

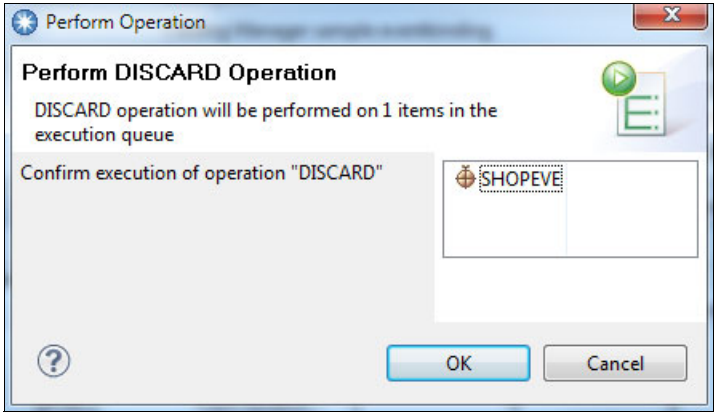


Figure 5-18 Confirm discard request

Important: When you disable an event binding, the bundle state changes to disabled. If you try to discard a disabled bundle resource when enabled resources that belong to the bundle are in the CICS region, CICS issues a message and the discard fails, as show in Figure 5-19. You must disable each of the enabled resources before the bundle resource is discarded. You can right-click **Disable** a bundle to disable all of its associated resources.

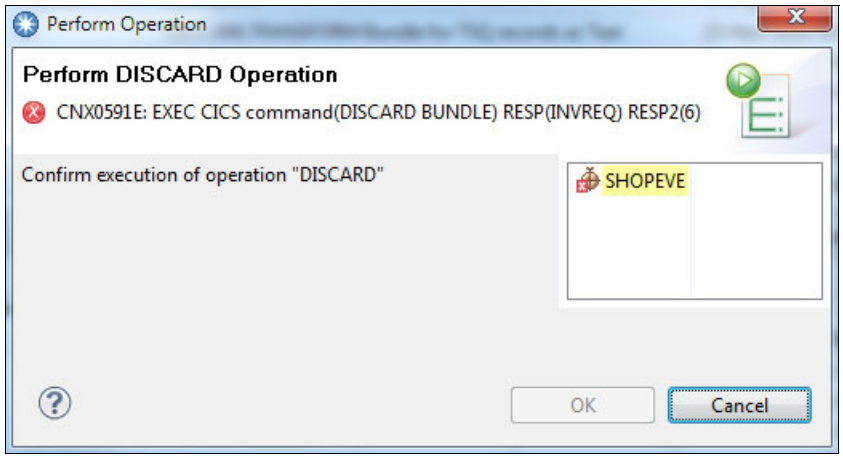


Figure 5-19 Discard of bundle fails

5.4.1 Replacing a deployed bundle

The following methods can be used to replace a deployed bundle after it is installed:

- Disable, discard, and then install the changed version of a bundle with the same name. No events are emitted from the moment the bundle is disabled until the moment the install completes successfully.
- You can replace a bundle without disabling the event binding by creating a new bundle, which must have a different name from the original bundle and contains the event binding with the same name. Events continue to be emitted until the bundle install completes successfully, at which point the new binding replaces the previous version.

5.5 Security considerations for CICS events

New security-related resources were added to support event processing in CICS TS. This section describes these new resources and shows how to set up CICS and the External Security Manager (ESM), in this case RACF, to protect event processing resources from unauthorized access.

5.5.1 Changes to security

Resource and command security apply to all event processing resources when these functions are enabled for the CICS region.

New category 1 transactions

The following transactions are category 1:

- CEPD: The event dispatcher, which is implemented by DFHEPDS.
- CEPM: The event queue manager, which is implemented by DFHEPSY.
- CEPF: The deferred filtering task for system events, which is implemented by DFHECDF.
- CRLR: The bundle resource resolver, which is implemented by DFHRLR.

These transactions are defined internally.

Resource security

Resource security for event processing resources uses resource profiles in the RCICSRES class, the WCICSRES grouping class, or equivalent customer-defined classes that are specified in the XRES system initialization parameter.

You must supply the following prefixes:

- EVENTBINDING to the name of the EVENTBINDING resource definition.
- EPADAPTER to the name of the EPADAPTER resource definition.
- EPADAPTERSET to the name of the EPADAPTERSET resource definition.
- CAPDATAPRED to the name of the CAPDATAPRED resource definition.
- CAPINFOSRCE to the name of the CAPINFOSRCE resource definition.
- CAPOPTPRED to the name of the CAPOPTPRED resource definition.
- CAPTURESPEC to the name of the CAPTURESPEC resource definition.

An example of this is shown in Example 5-4 on page 102.

Command security

Command security for event processing resources uses the following resources in the CCICSCMD class or the VCICSCMD grouping class:

- EVENTPROCESS
- EVENTBINDING
- EPADAPTER
- EPADAPTERSET
- CAPDATAPRED
- CAPINFOSRCE
- CAPOPTPRED

Security that uses the XRES resource security parameter

The XRES system initialization parameter is used to security check CICS resources.

CICS profiles are passed to the security manager for checking. For more information, see CICS Information Center at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v5r1/topic/com.ibm.cics.ts.doc/dfht5/topics/dfht5_doct.html

5.5.2 Setting up CICS security for event bindings

This section shows how to use an ESM (in this case, RACF) to protect access to the eventbinding resource.

The CMAS in this example has simulated CICS Resource Security enabled.

In the SIT for CICS EPREDA01, the security options are set as shown in Example 5-3. The XRES option is the specific option that is needed to enable resource checking.

Example 5-3 CICS SIT security options

```
SEC=YES
XAPPC=NO
XCMD=NO
XDB2=NO
XDCT=NO
XEJB=NO
XFCT=NO
XJCT=NO
XPCT=NO
XPPT=NO
XPSB=NO
```

```
XRES=YES
XTRAN=NO
XTST=NO
```

RACF definitions

This example shows how to protect the BUNDLE, which is named SHOPEVE. This bundle contains an EVENTBINDING called ShoppingEventBinding.

From the TSO command prompt, run the commands that are shown in Example 5-4to define profiles in RACF for the BUNDLE and EVENTBINDING in the RCICSRES class. This permits the user ID `CICSRS7` to use these profiles.

Example 5-4 RACF Permissions for BUNDLE and EVENTBINDING

```
RDEFINE RCICSRES   BUNDLE.SHOPEVE UACC(NONE))
PERMIT BUNDLE.SHOPEVE CLASS(RCICSRES)   ID(CICSRS7)   ACCESS(ALTER)
RDEFINE RCICSRES   EVENTBINDING.ShoppingEventBinding UACC(NONE)
PERMIT EVENTBINDING.ShoppingEventBinding CLASS(RCICSRES)   ID(CICSRS7)
ACCESS(ALTER)
```

Important: RACF profile definitions and security checking are case-sensitive.

Run the **`RACF SETROPTS`** command to refresh the `RCICSRES` class. Specify `ALTER` access above as this is needed to allow the user ID `CICSRS7` to DISCARD an event binding or bundle.

With these options set, the CICS user ID `CICSRS7` can see bundle SHOPBUN and event binding ShoppingEventBinding by using the CICS Explorer. This user ID also can INSTALL a bundle definition and DISABLE or DISCARD this event binding and bundle.

A user ID other than `CICSRS7` cannot see the event bindings or bundles that are installed in the CICS region. An attempt to access these resources does not generate a security error message in the CICS Explorer. This is a feature of how CICS works. To help explain this feature, compare the behavior for an unauthorized user of the CICS Explorer with the way in which `CEMT INQUIRE EVENTBINDING` works.

For example, by using `CEMT INQUIRE EVENTBINDING(*)` with generic target generates a `NOTFOUND` response, even when the user ID that is running the command is not authorized to the EVENTBINDING profile, as shown in Example 5-5 on page 103.

Example 5-5 CEMT INQUIRE EVENTBINDING: Generic

```
I EVENTBINDING(*)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Eventb(*                   )                    NOT FOUND




                                                    SYSID=RED7
APPLID=EPREDA01
 RESPONSE: 1 ERROR TIME: 06.30.15  DATE: 07/14/09
```

Whereas, a `CEMT INQUIRE EVENTBINDING` that specifies the event binding name receives a `NOT AUTHORIZED` response, as shown in Example 5-6.

Example 5-6 CEMT INQ EVENTBINDING - specific

```
I EVENTB(SHOPPINGEVENTBINDING)
 STATUS:  RESULTS - OVERTYPE TO MODIFY
  Eventb(SHOPPINGEVENTBINDING )                  NOT AUTHORIZED




                                                     SYSID=RED7
APPLID=EPREDA01
  RESPONSE: 1 ERROR                             TIME: 09.35.53  DATE:
07/10/09
PF 1 HELP       3 END       5 VAR        7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

In the CICS job log, we see the security violation message that shows that the CICS default user `CICSUSER` is not authorized, as shown in Example 5-7.

Example 5-7 CICS job log showing security violation

```
DFHXS1111 07/14/2009 09:35:53 EPREDA01 CEMT Security violation by user
CICSUSER for resource EVENTBINDING.SHOPPINGEVENTBINDING in class
RCICSRES. SAF codes are (X'00000008',X'00000000'). ESM codes are
(X'00000008',X'00000000').
```

Because the CICS Explorer obtains all of the attributes for all event bindings and then applies a filter to the results, it does not generate a security violation.

For example, if a user that is signed on to the CICS Explorer attempts to see event bindings for which they are not authorized, the event bindings are not displayed.

Details of the CICS Explorer signed-on user ID and password are entered in the CICS Explorer connection panel.

Important: To install a bundle resource (BUNDDEF) from the CICS Explorer that is connected to CICSPlex SM WUI and uses CICSPlex simulated resource security (as in this case), a CICSPlex APAR PK91529 is required.

5.5.3 The user ID in EP adapters

EP adapters can emit events by using the user ID from the security context at the time of event capture or you can specify an ID in the adapter that is used for event emission. It is useful to specify an ID when you do not want to grant permission to the application's task ID to use the event emission transport but you still want to emit events.

When you install a BUNDLE resource that includes an EP adapter for which you specified a user ID, CICS checks that the user ID that is performing the install operation is authorized as a surrogate user of the specified user ID. This check also applies to the CICS region user ID during group list install on a CICS cold or initial start.

This example specifies a user ID `CICSRS9` in the EP adapter, as shown in Figure 5-20 on page 105. It also shows how to set up the security authorization that is required to install the bundle that contains this EP adapter during an initial start of CICS with an RDO definition that is added to the CICS group list.

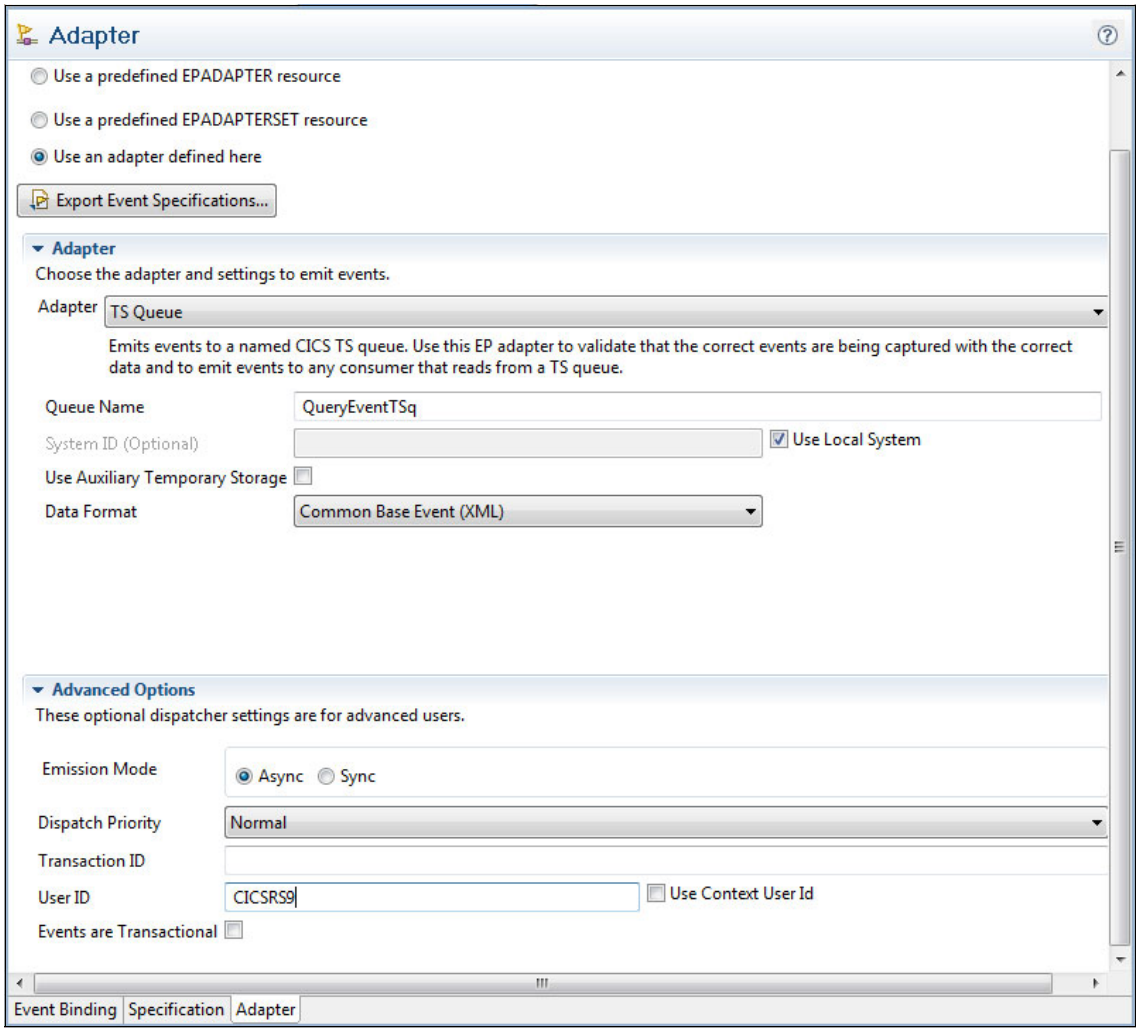


Figure 5-20 Adapter with user ID specified

With user ID `CICSRS9` set in the EP adapter, you can attempt to INITIAL start CICS and install the bundle SHOPEVE through an RDO group `SHOPBUND` that is included in the CICS group list. The install of the event binding `ShoppingEventBinding` now fails with a security violation as expected, as shown in Example 5-8 on page 106.

Example 5-8 Security violation

```
DFHRL0107 I 07/15/2009 12:48:38 EPREDA01 CICSRS7 The CICS resource life-cycle manager
has started to create the BUNDLE resource SHOPEVE.
DFHXS1111 07/15/2009 12:48:38 EPREDA01 CSSY Security violation by user CICSRS7 for
resource CICSRS9.DFHINSTL in class SURROGAT. SAF codes are (X'00000004',X'00000000').
ESM codes are (X'00000004',X'00000000').
DFHEC1010 07/15/2009 12:48:38 EPREDA01 Userid CICSRS7 is not authorized to create
EVENTBINDING ShoppingEventBinding with an EP adapter userid of CICSRS9.
DFHRL0102 E 07/15/2009 12:48:38 EPREDA01 CSSY The CICS resource life-cycle manager
failed to create the resource ShoppingEventBinding
      and returned with reason CALL_BACK_ERROR.
DFHAM4893 I 07/15/2009 12:48:38 EPREDA01 Install for group SHOPBUND has completed
successfully.
```

Now issue the RACF commands in TSO to allow the install to work, as shown in Example 5-9.

Example 5-9 RACF command

```
RDEFINE SURROGAT CICSRS9.DFHINSTL UACC(NONE) OWNER(CICSRS9)
PERMIT CICSRS9.DFHINSTL CLASS(SURROGAT) ID(CICSRS7) ACCESS(READ)
```

Issue RACF SETROPTS to refresh the SURROGAT class profile and then try to INITIAL start CICS. This time it is successful, as shown in Example 5-10.

Example 5-10 Initial start of CICS

```
DFHRL0107 I 07/15/2009 12:58:14 EPREDA01 CICSRS7 The CICS resource life-cycle manager
has started to create the BUNDLE resource
       SHOPEVE.
DFHEC1001 07/15/2009 12:58:14 EPREDA01 Event binding ShoppingEventBinding installed
successfully.
DFHRL0109 I 07/15/2009 12:58:14 EPREDA01 CSSY The CICS resource life-cycle manager
has created the BUNDLE resource SHOPEVE and the
      BUNDLE is in the enabled state.
DFHAM4893 I 07/15/2009 12:58:14 EPREDA01 Install for group SHOPBUND has completed
successfully.
```

6

Capturing application events

This chapter shows how to use the CICS Explorer to create, deploy, and verify an example event that is captured from a CICS application by using the shopping scenario.

This chapter includes the following topics:

- Create the CICS bundle project
- Creating the event binding
- Creating the TS Queue adapter
- Exporting the CICS bundle project
- Installing the CICS bundle
- Testing the shopping application
- Completing the shopping scenario

The steps that are described in these topics can be used as a basis to create the event bindings and event adapters that are used in the shopping scenario.

6.1 Create the CICS bundle project

Complete the following steps to create a CICS bundle project by using the CICS Explorer:

1. Switch to the Resource perspective.
2. Click the **Create CICS Bundle Project** icon, as shown in Figure 6-1. Alternatively, select **File** → **New Wizards** → **Other** → **CICS Resources** → **CICIS Bundle Project** → **Next**.

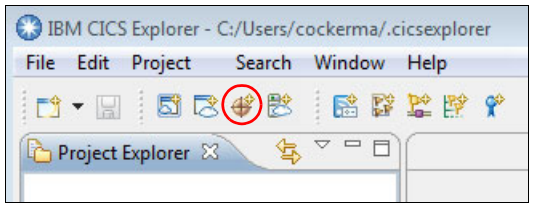


Figure 6-1 Starting the Create Bundle Project wizard

3. Enter the project name `ShoppingEventBundle`, as shown in Figure 6-2, then click **Finish**. This project contains an event binding and event adapter for the shopping application.

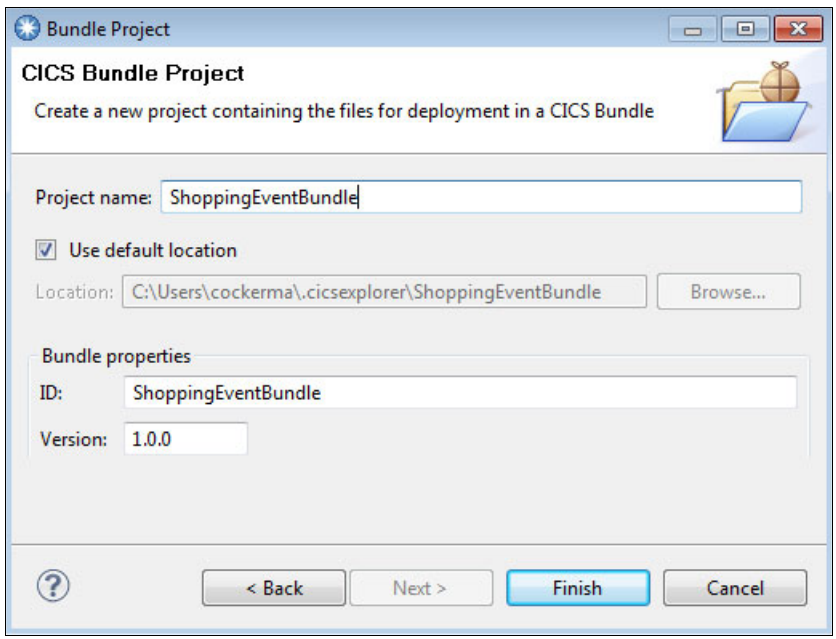


Figure 6-2 Creating the ShoppingEventBundle bundle project

6.2 Creating the event binding

Complete the following steps to create the event binding:

1. In the Project Explorer view, right-click the **ShoppingEventBundle** project, then select **New** → **CICS Event Binding**.
2. Enter the file name `ShoppingEventBinding`, as shown in Figure 6-3, then click **Finish**. The event binding editor starts automatically.

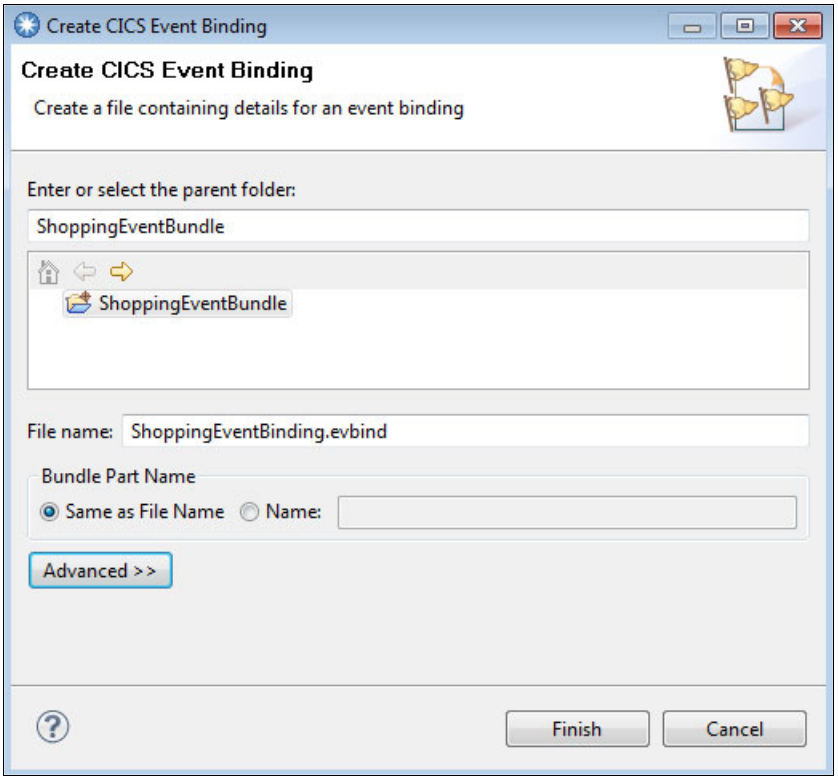


Figure 6-3 Creating the ShoppingEventBinding event binding

Important: An error appears in the Problems view (see Figure 6-4 on page 110) because the event binding does not yet have an event specification. The event binding and CICS bundle also have error markers (a red box with a white X) to denote there is a problem. This error is resolved after the event specification is added.

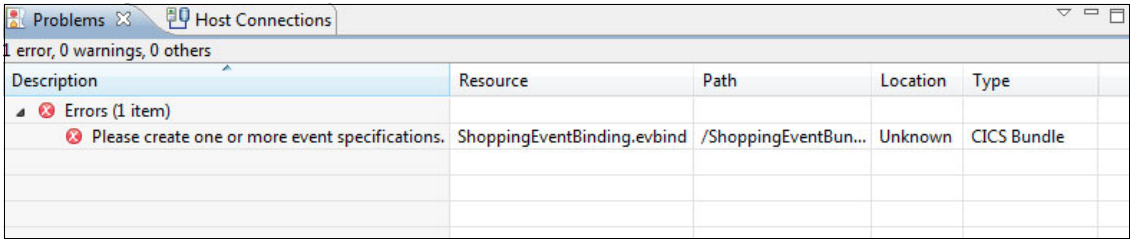


Figure 6-4 Problem shown after creating an event binding

3. In the event binding editor in the Event Binding tab, enter the description `The Shopping Event Binding`, as shown in Figure 6-5.

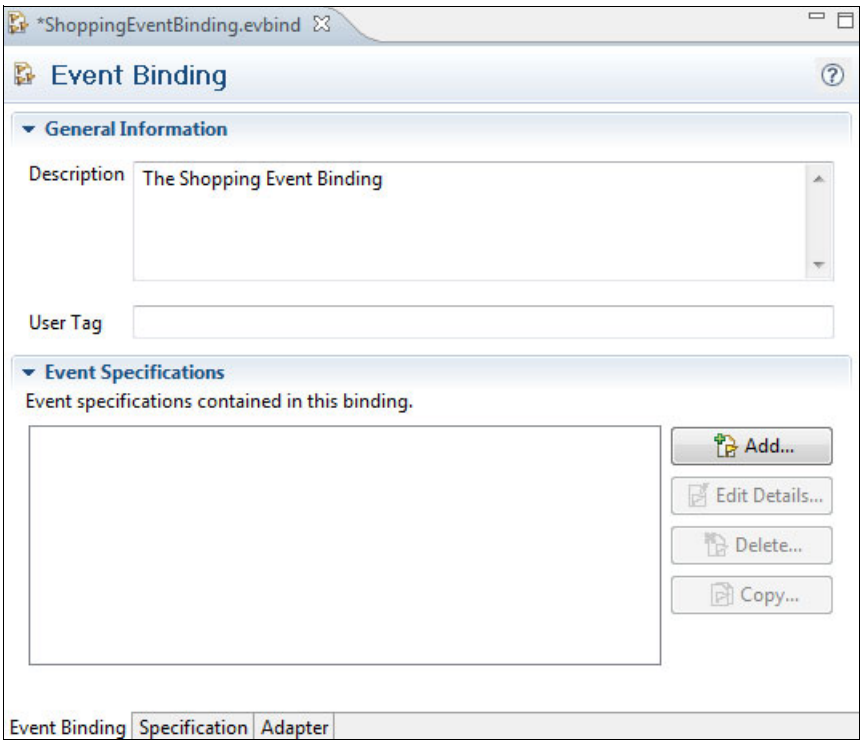


Figure 6-5 Adding the QueryStock binding specification

4. Click **Add** to add an event specification, and enter the name `QueryStock` and description `Query stock event specification`, as shown in Figure 6-6 on page 111. Click **OK**.

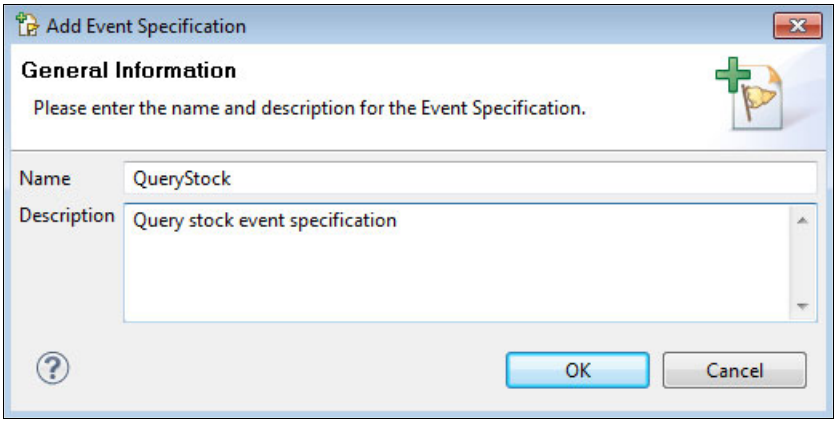


Figure 6-6 Naming the QueryStock event specification

5. Click **Edit Details** to edit the event specification.
6. To emit the customer number and stock identification with the event, they must be added in the Emitted Business Information section. Click **Add** to add the first information item.
7. The Emitted Business Information window opens, as shown in Figure 6-7 on page 112. Enter the following values and click **OK**:
   - Name: `CustomerNumber`
   - Type: `Numeric`
   - Length: `6`
   - Precision: 0
   - Description: `The customer number`

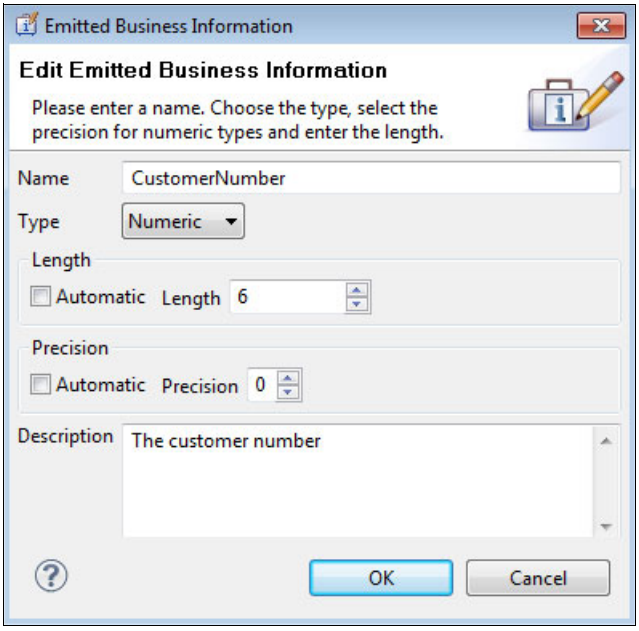


Figure 6-7 Adding the CustomerNumber business information

8. Select **Add** again, and in the Emitted Business Information window, enter the following values and click **OK**:
   - Name: `StockId`
   - Type: `Numeric`
   - Length: `6`
   - Precision: 0
   - Description: `The stock identification`
9. In the Specifications tab shown in Figure 6-8 on page 113, select **Add a Capture Specification**. This is where to indicate when and how an event should be captured from the application.

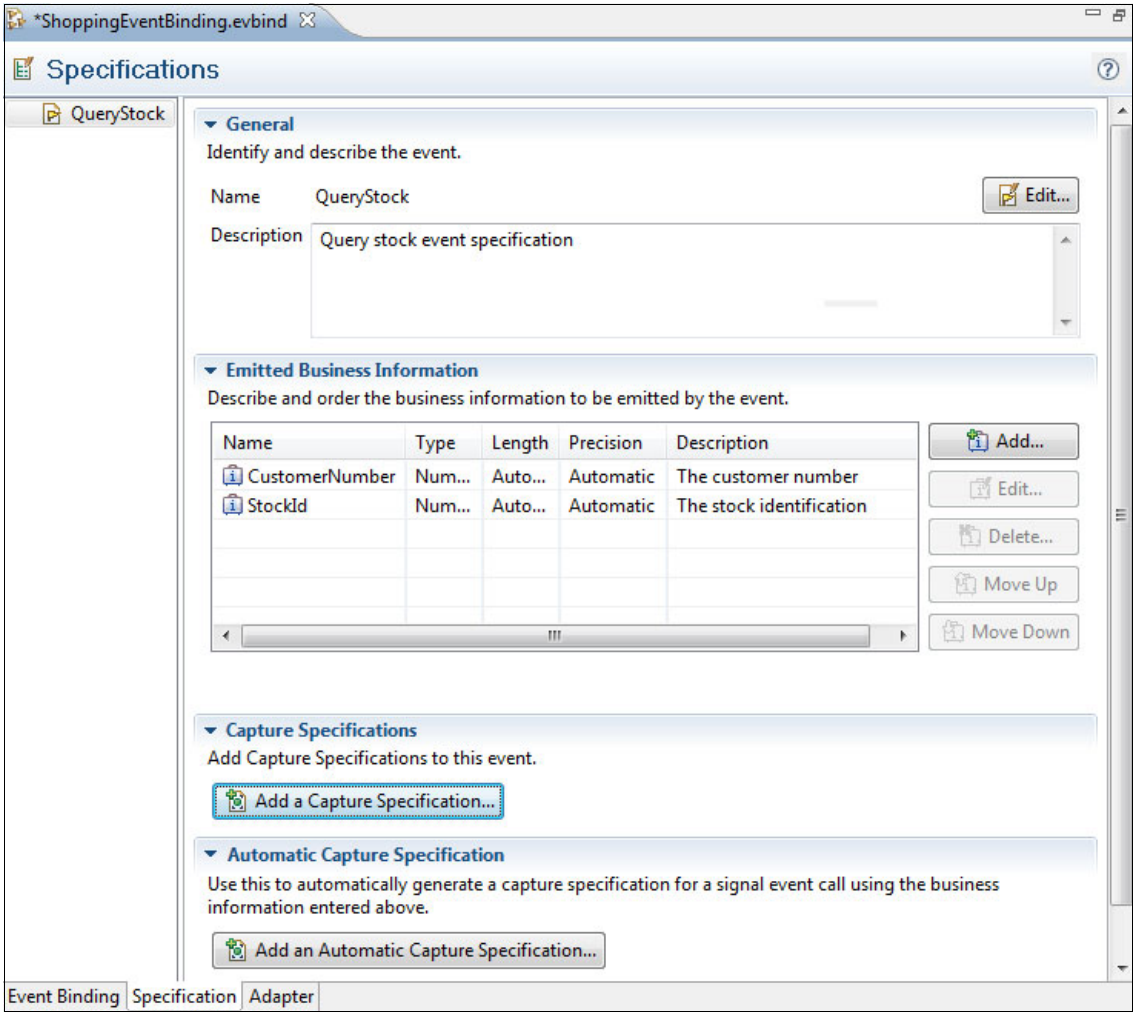


Figure 6-8 Adding the CaptureQueryEvent capture specification

10. In the Add Capture Specification window, enter the following values and click **OK**:
    - Name: `CaptureQueryEvent`
    - Description: `Capture the Query Event`
11. Select **CaptureQueryEvent** in the tree on the left side. The editor shows the capture specification, including the tabs Capture Point, Filtering, and Information Sources.

12. In the Capture Point section, select **PROGRAM INIT**, as shown in Figure 6-9. This captures the event when the program is initialized by CICS, and before the program is run.

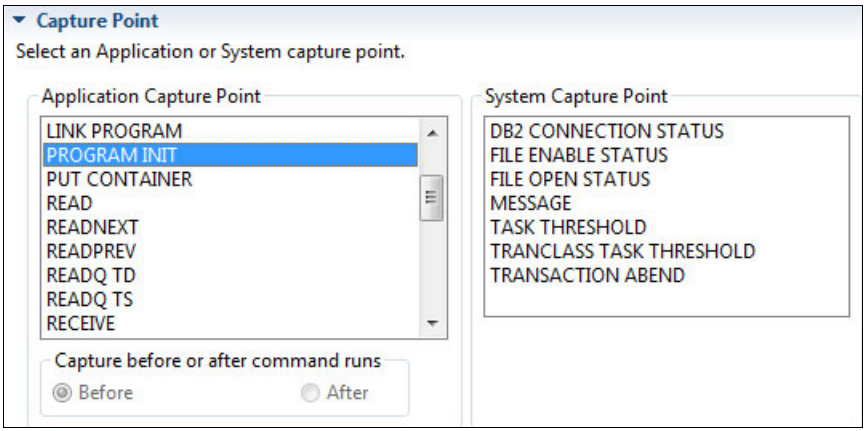


Figure 6-9 Selecting the PROGRAM INIT capture point

13. Select the Filtering tab. The Application Event window opens.
14. Set the program name predicate by setting the PROGRAM operator to `Equals`, and Value to `QUERY` in the Event Options section, as shown in Figure 6-10 on page 115. This filters the Program initiation that is performed by CICS to capture only those where the program name is QUERY.

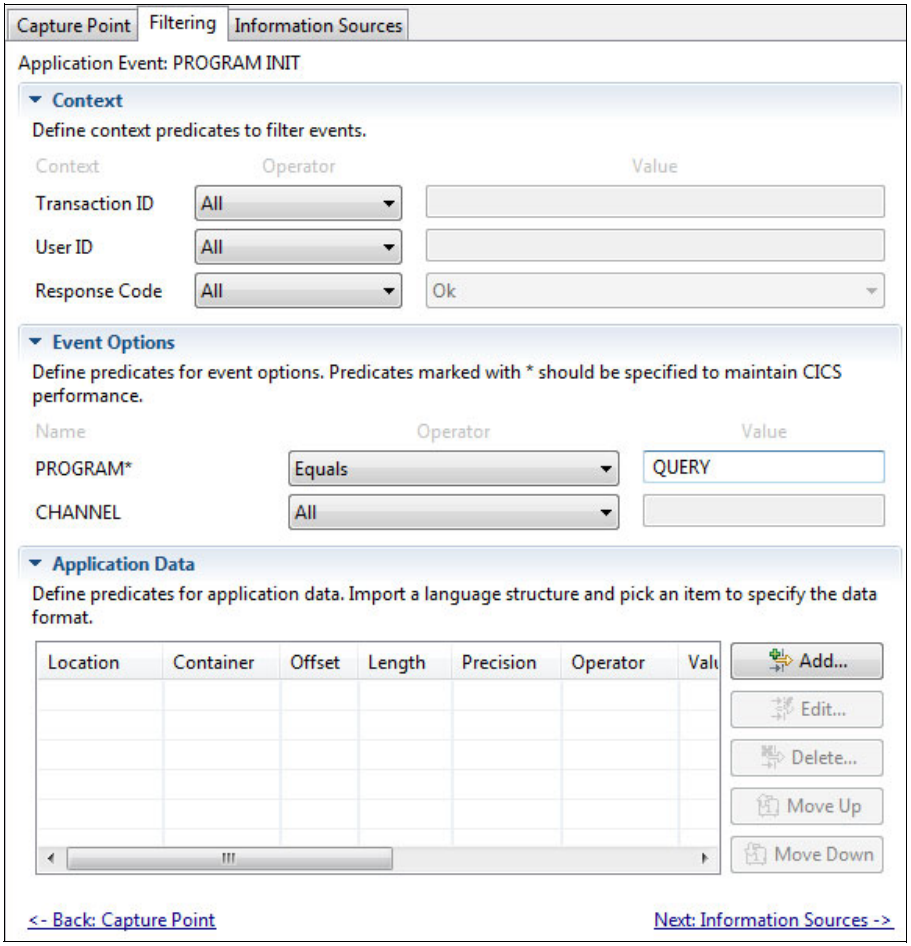


Figure 6-10 Adding the application context predicates

15. When the QUERY program is initiated it expects the DISP-STOCK-ITEM container to exists for stock inquiries. To create a predicate for this situation, select **Add** in the **Application Data** section.

16. In the Application Data Predicate window, enter the following values as shown in Figure 6-11 on page 116 and click **OK**:

    – Operator: `Exists`
    – Location: `CHANNEL`
    – Container: `DISP-STOCK-ITEM`

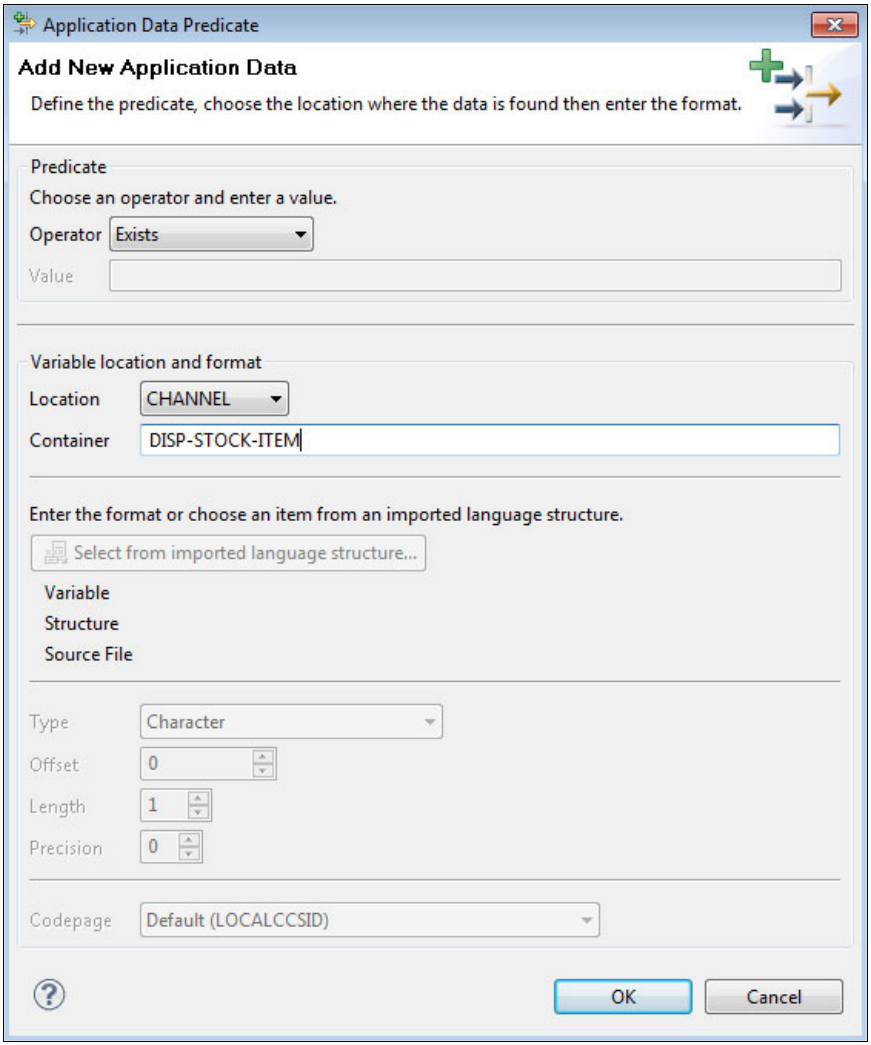


Figure 6-11 Adding the DISP-STOCK-ITEM container application data predicate

Tip: Application data predicates are evaluated by CICS in the order they are defined here. For best performance, place those predicates that filter out the largest number of unwanted events before others. Use the Move Up and Move Down buttons to change the order.

17. Select the Information Sources tab to define how CICS should capture the data that is required by the emitted business information that is shown in Figure 6-12 as `CustomerNumber` and `StockId`. The table is pre-filled with the items that were specified in the Emitted Business Information section of the Event Binding Editor.

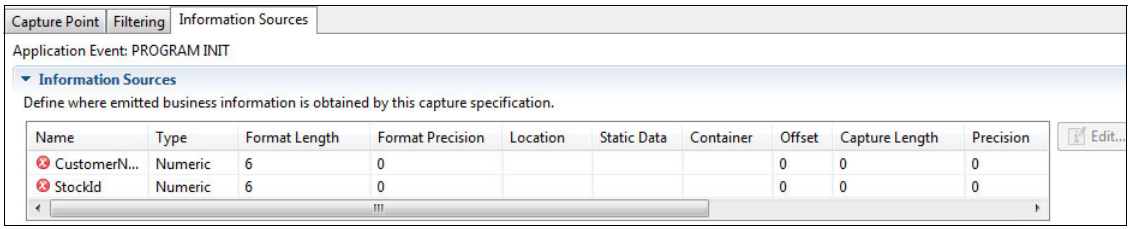


Figure 6-12 Defining where emitted business information items should be captured from

18. Select the `CustomerNumber` item, then click **Edit**.
19. In the Information Source for CustomerNumber window, select **CHANNEL**. Enter the following values as shown in Figure 6-13 on page 118 and click **OK**:
    - Container: `CUSTOMER-NO`
    - Type: `Packed Decimal`
    - Offset: `0`
    - Length: `3`
    - Precision: `0`

> **Important:** The emitted business information source is stored as a COBOL packed decimal data type and has a length of three bytes, whereas the emitted business information is string format and has a length of six bytes. CICS converts data between these data types.

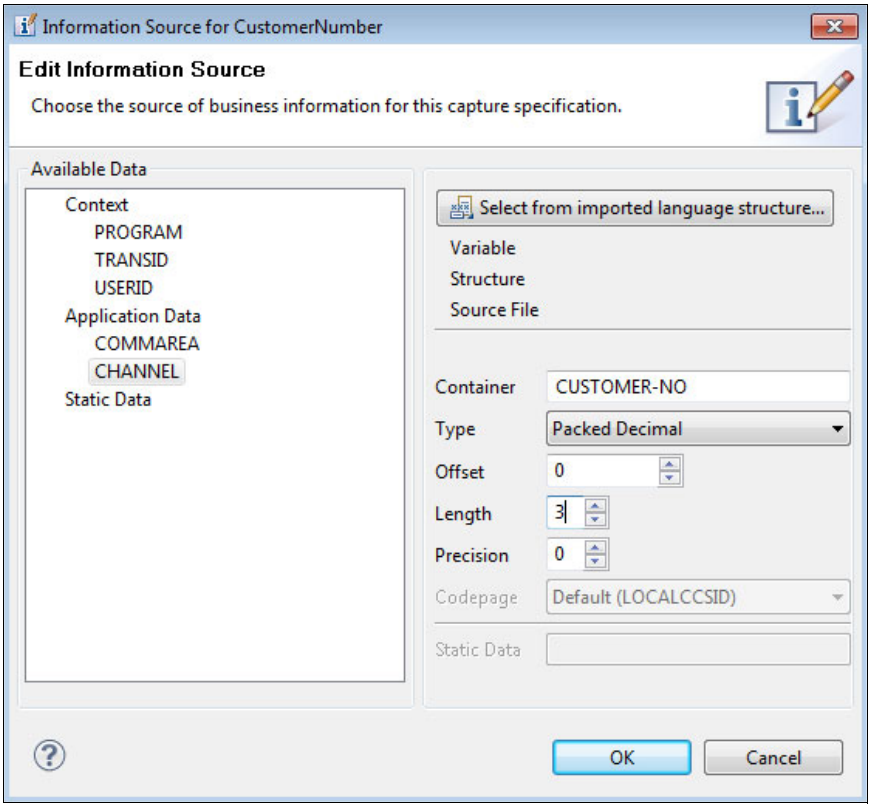


Figure 6-13 Defining the information source for customerNumber

Tip: If you have the copy book describe the data in the container, you can use **Select from imported language structure** to import the copy book and select the field. The type, offset, length, and precision is then automatically set. In addition, the variable name, structure name, and file name for the imported field is recorded for later use by the CICS Explorer search facility.

20. Repeat the previous step for `StockId` with the following values and click **OK**:
    - Container: `DISP-STOCK-ITEM`
    - Type: `Packed Decimal`
    - Offset: `0`
    - Length: `3`
    - Precision: `0`

The event specification is now complete.

6.2.1 Referencing the TS Queue adapter

For test purposes, the event is emitted to a temporary storage queue by using the TS Queue EP adapter. This EP adapter is useful to quickly verify that only the wanted events are emitted; for example, the capture point and filtering are correctly defined. It is also useful to ensure the event contains the correct information; for example, the emitted business information items, information sources, and formatting are correctly defined.

Rather than define the EP adapter in the event binding, the following steps add a reference to an independent EP adapter that is defined in 6.3, “Creating the TS Queue adapter” on page 120. This enables the EP adapter to be reused and the event binding to target a different EP adapter after initial testing is complete:

1. Select the Adapter tab in the event binding editor.
2. Select **Use a predefined EPADAPTER resource**.
3. Enter the name `TSQ`, as shown in Figure 6-14.

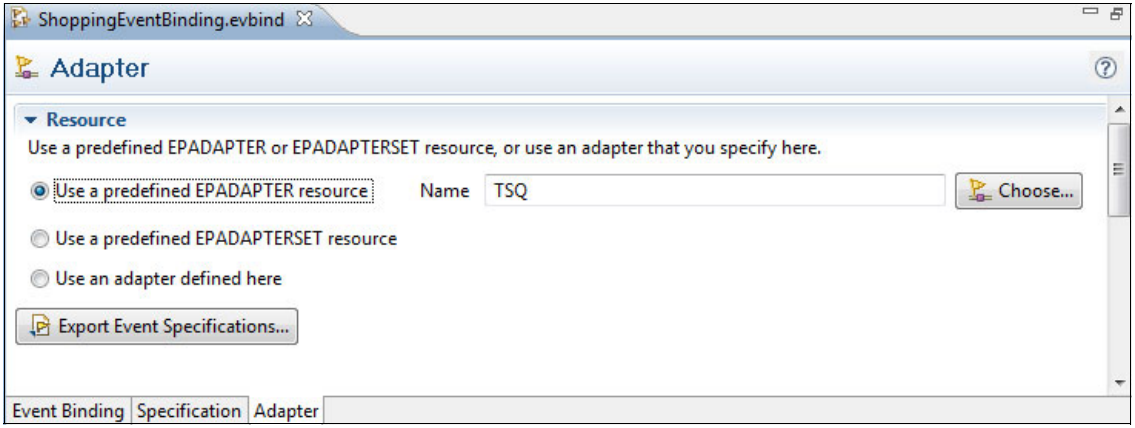


Figure 6-14 Adding a reference to the EP adapter that will emit the event

4. Select **File** → **Save**, or press Ctrl+S to save the event binding.

6.3 Creating the TS Queue adapter

Complete the following steps to create the TS Queue adapter:

1. In the Project Explorer view, right-click the **ShoppingEventBundle** project then select **New** → **CICS Event Processing Adapter**.
2. Enter the File name `TSQ`, then select **Finish**.
3. In the Adapter section, enter the following information, as shown in Figure 6-15:
   - Description: `Emit events to the temporary storage queue QueryEventTSQ in XML format.`
   - For Adapter, select **TS Queue**.
   - Queue Name: `QueryEventTSQ`
   - Data Format: `WebSphere Business Events (XML)`

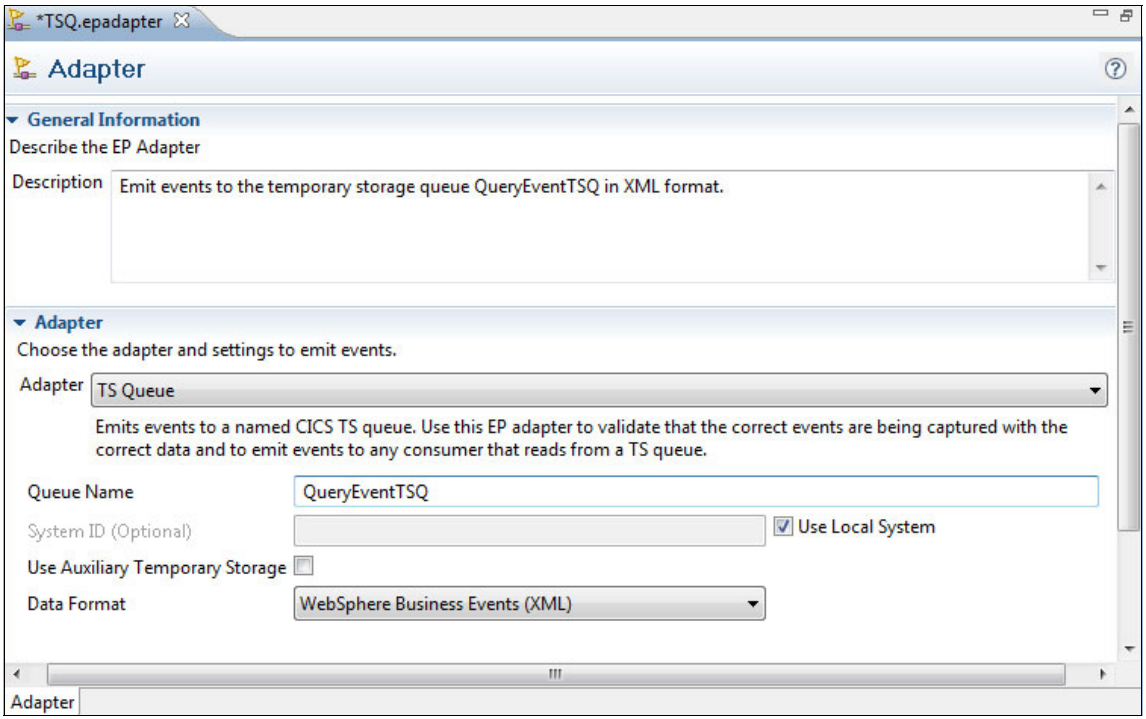


Figure 6-15 Specifying the TS Queue event adapter

4. Select **File** → **Save** to save the event adapter.

6.3.1 Exporting the event specification

The event is emitted in the data format that is specified in the event adapter; in this case, the format is WebSphere Business Events (XML).

Complete the following steps to create the schema for this XML document. The schema can be imported by IBM Operational Decision Manager tooling so it understands what information is in the events it uses:

1. Create a directory on your workstation for the event specifications, such as `C:\ExportedEventSpecifications`.
2. Double-click the `ShoppingEventBinding.evbind` file to open the event binding editor.
3. Select the Adapter tab.
4. In the Resource section, select **Export Event Specification**.
5. Enter the directory `C:\ExportedEventSpecifications`, as shown in Figure 6-16, then select **OK**.

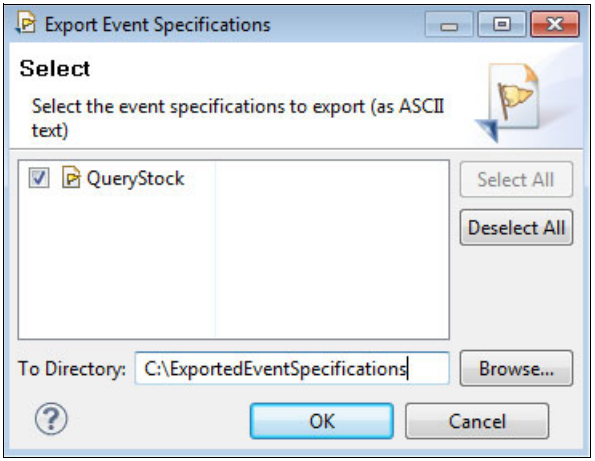


Figure 6-16 Exporting the schema for the event

6. A window opens that confirms the `QueryStock.xsd` file was created. Select **OK**. The contents of this file are shown in Example 6-1 on page 122. Note the CustomerNumber and StockId elements and several others that CICS automatically includes.

Example 6-1 QueryStock.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="qualified"
            targetNamespace="http://cics.ibm.com/QueryStock"
xmlns:tns="http://wbe.ibm.com/6.2/Event/QueryStock">
  <xsd:element name="QueryStock">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" minOccurs="0"
name="QueryStock_Context">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element minOccurs="0" maxOccurs="1" name="Binding user
tag" type="xsd:string" />
              <xsd:element minOccurs="0" maxOccurs="1" name="Network UOWID"
type="xsd:string" />
              <xsd:element minOccurs="0" maxOccurs="1" name="businessevent"
type="xsd:string" />
              <xsd:element minOccurs="0" maxOccurs="1" name="Capture Spec
Name" type="xsd:string" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element maxOccurs="unbounded" minOccurs="0"
name="QueryStock_Data">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element minOccurs="0" maxOccurs="1"
name="CustomerNumber" type="xsd:decimal" />
              <xsd:element minOccurs="0" maxOccurs="1" name="StockId"
type="xsd:decimal" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

7. You can close the event binding editor by right-clicking the Editor tab, or pressing Ctrl-W.

Important: CICS also supports other XML schemas and the CICS Flattened Event format (a non-XML binary format). The CICS Flattened Event is a language structure that is suitable when the event is going to be used by a program that is written in COBOL, C, or PL/I.

6.4 Exporting the CICS bundle project

For CICS to access the bundle project, it first must be exported from the Eclipse workspace on your workstation to a directory on zFS. Complete the following steps to export the project:

1. In the Project Explorer view, right-click the **ShoppingEventBundle** project and click **Export Bundle Project to z/OS UNIX File System,** as shown in Figure 6-17.

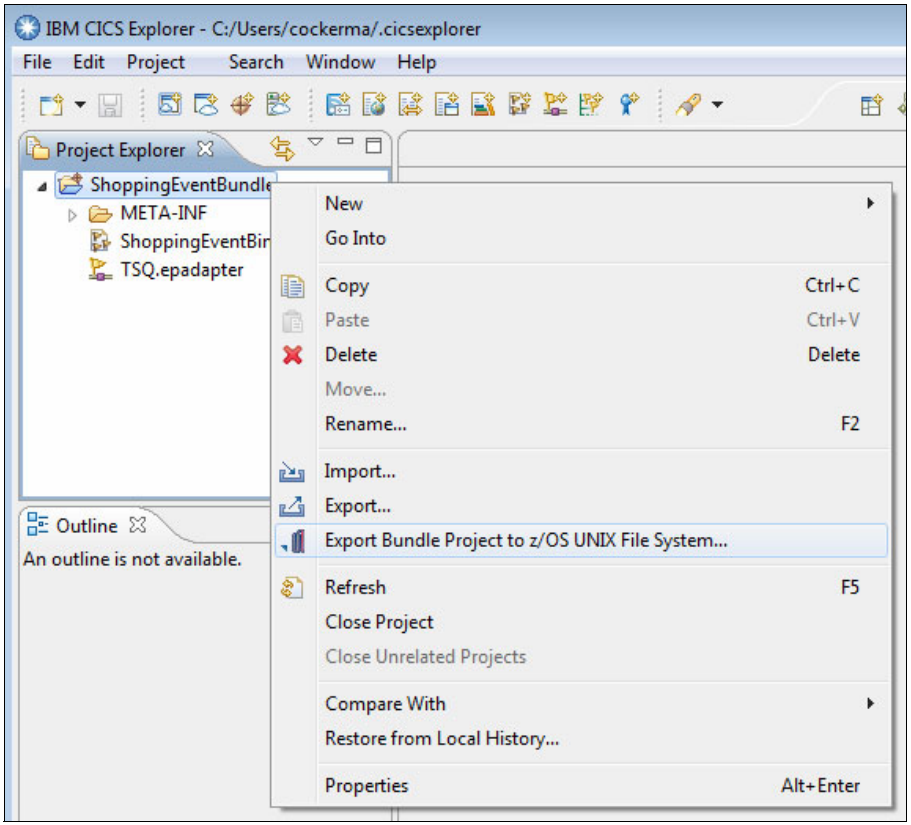


Figure 6-17 Starting the export bundle project wizard

2. In the Export to z/OS UNIX File System window, select **Export to a specific location in the file system**, then select **Next**.
3. Select a suitable connection; for example, `wtsc66:21` as shown in Figure 6-18 on page 124. If you have not recently signed on, the Signon window opens. Enter your z/OS user ID and password, then click **OK**.

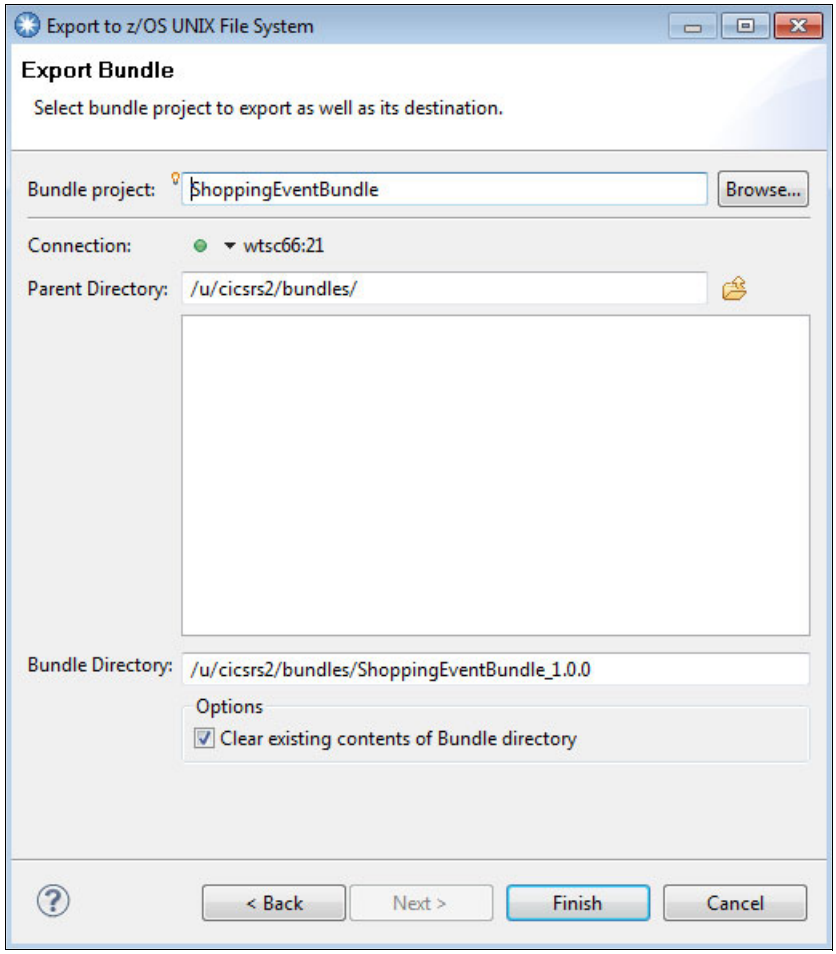


Figure 6-18 Exporting the bundle project ShoppingEventBundle to zFS

4. Enter the parent directory `/u/cicsrs2/bundles`, or use the tree to browse to it.
5. Select **Clear existing contents of Bundle directory** to remove existing files in the parent directory.
6. Click **Finish**.
7. To view the exported project files, switch to the **z/OS** perspective and display the z/OS UNIX FIles view. Change the path to `/u/cicsrs2/bundles`, as shown in Figure 6-19 on page 125.

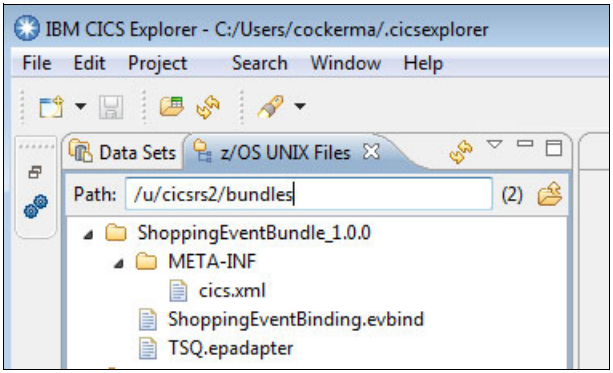


Figure 6-19 Exported ShoppingEventBundle bundle

6.5 Installing the CICS bundle

Complete the following steps to install the new version of the CICS bundle into the CICS system by creating and installing a CICS BUNDLE resource:

1. In CICS Explorer, switch to the CICS SM perspective.
2. In the CICSplex Explorer view, select **EPREDA01**, which is the applid of the CICS system. This sets the scope for other views in this perspective.
3. To view the installed BUNDLE definitions, select **Operations** → **Bundles**. The Bundles view opens.
4. If the SHOPEVE resource is shown, you must disable and then discard it as follows:
   a. Right-click the SHOPEVE bundle, then select **Disable** → **OK.**
   b. Right-click the SHOPEVE bundle, then select **Discard** → **OK.**
5. To install the new SHOPEVE bundle, select **Definitions** → **Bundle Definitions**. The Bundles view opens.
6. Define the SHOPEVE bundle, as described in 5.3.1, “Creating a new CICS bundle definition” on page 89.
7. Right-click the SHOPEVE bundle, then select **Install**. Select the CICS region **EPREDA01**, then click **OK**.

8. To verify the state of the bundle, select **Operations** → **Bundles**. Figure 6-20 shows the installed SHOPEVE bundle resource.

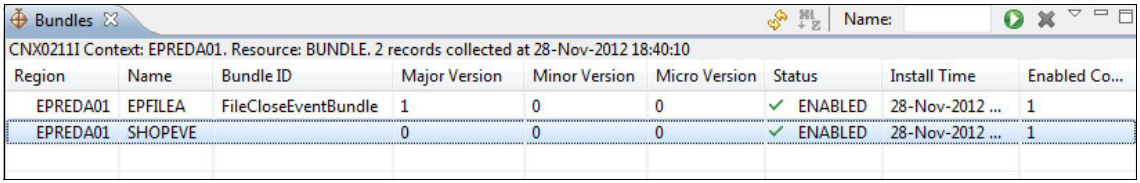


Figure 6-20 Bundles view to show installed BUNDLE resources

6.6 Testing the shopping application

You can test to see whether a customer query on a stock item in the shopping application causes a QueryStock event to be emitted.

Complete the following steps to test the shopping application:

1. On a CICS terminal, enter the transaction ID `MENU` and press Enter. Enter the customer number `00005`, as shown in Figure 6-21. Press PF1 to query an item.

```
Order application

Customer Number: . . . . . 00005

PF1    Query Item
PF2     Order Item
PF4    fulfill Order
PF5    Ship Order

     Exit = PF3
```

Figure 6-21 Shopping application MENU transaction to query stock

2. Enter S before ID 00006, as shown in Figure 6-22, and press Enter.

```
Query List
Customer Number: 00005

  S ID    Description
  - ----- ------------------------------------
    00001 STOCK1
    00002 STOCK2
    00003 STOCK3
    00004 STOCK4
    00005 STOCK5
  S 00006 STOCK6
```

Figure 6-22 Selecting stock item to query

The result is shown in Figure 6-23.

```
Query Item
Customer Number: 00005


             Stock ID: 00006

          Description: STOCK6

                Value:     6.99

          Stock Level: 00600
```

Figure 6-23 Query stock item result

6.6.1 Verifying the event was emitted

Complete the following steps to verify that the event was captured and emitted to the QueryEventTSQ:

1. To show the event was captured, open the Capture Specifications view by selecting **Operations** → **Event Processing** → **Capture Specifications**.

   For the CaptureQueryEvent entry, the Events Captured column has a value of 1, as shown in Figure 6-24 on page 128.

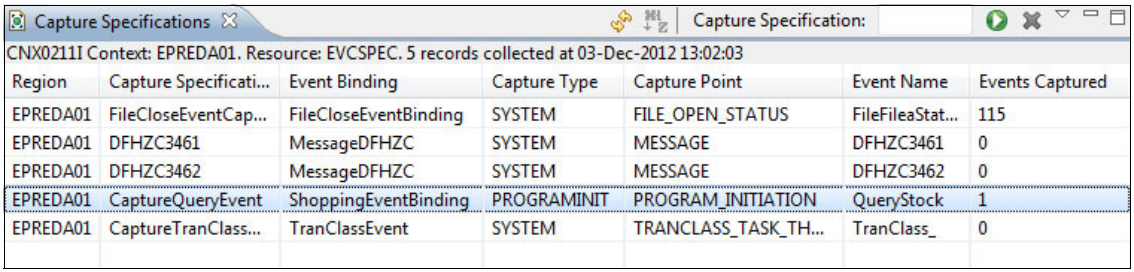


Figure 6-24 Verifying the number of captured events

2. To show that an event was emitted, open the TS Queues window by selecting **Operations** → **Queues** → **TS**. For the QueryEventTSQ entry, the Item Count column has a value of 1, as shown in Figure 6-25.

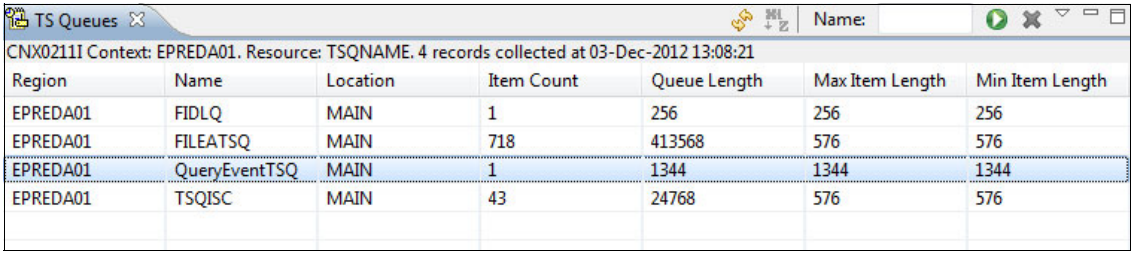


Figure 6-25 Verifying the number of emitted events on the TS queue

3. To display the contents of QueryEventTSQ, log on to CICS and use the **`CEBR QueryEventTSQ`** command.

> **Important:** This CEBR command requires the terminal to have automatic uppercase translation turned off. This can be done with the **`CEOT NOUCTRAN`** command.

4. The output from CEBR is shown in Example 6-2. The text ".." is substituted for the carriage return and line feed characters. Press PF9 to scroll right to view the rest of the event.

Example 6-2 CECI READQ TS command

```
CEBR  TSQ QueryEventTSQ    SYSID EA01 REC     1 OF     1    COL     1 OF
677
 ENTER COMMAND ===>
      *************************  TOP OF QUEUE
*****************************
00001 <?xml version="1.0" standalone="yes"?>..<connector
name="ShoppingEventBin
      ************************  BOTTOM OF QUEUE
*****************************
```

6.7 Completing the shopping scenario

So far in this chapter, the ShoppingEventBundle contains the `ShoppingEventBinding.evbind` and `TSQ.epadapter`. To complete the scenario that is described in 4.3, "Sample scenarios" on page 73, create the following event bindings and event adapters within the ShoppingEventBundle:

- `ShoppingEventBundle`
  - `QueryVsSale.evbind`
    - Contains an event capture specification for when a stock item is queried.
    - Contains an event capture specification for when an order is placed.
    - References the event adapter `wodm.epadapter` for event emission.
  - `LowStock.evbind`
    - Contains an event capture specification for when the stock level is low.
    - References the event adapter `BM.epadapter` for event emission.
  - `ShippedOrder.evbind`
    - Contains an event capture specification for when an order is shipped.
    - References the event adapter `BM.epadapter` for event emission.
  - `HighValueOrder.evbind`
    - Contains an event capture specification for when a high value order is shipped.
    - Contains an adapter specification that uses `BM.epadapter` for event emission.

- `WODM.epadapter`
  - An HTTP adapter to emit events to the IBM Operational Decision Manager system.
  - References the URIMAP `WODM.urimap`.
- `WODM.urimap`

  A URIMAP resource definition that specifies TCP/IP connection details to the IBM Operational Decision Manager system.
- `BM.epadapter`
  - An HTTP adapter to emit events to the IBM Business Monitor system.
  - References the URIMAP `BM.urimap`.
- `BM.urimap`

  A URIMAP resource definition that specifies TCP/IP connection details to the IBM Business Monitor system.
- `TSQ.epadapter`

  A TS queue adapter to emit events to a temporary storage queue for testing.

For more information, see Chapter 10, “IBM Business Monitor” on page 201.

7

Generating CICS system events

This chapter describes the steps that are needed to generate some sample simple events.

You can experiment to create a system event that writes a TS queue message when a file named STOCK is closed. The second project is more complex and by using it you can control the TRANCLASS TASK THRESHOLD condition and send an appropriate message to the console when it is reached.

This chapter includes the following topics:

- Creating a system event FileClose
- Sample system events for alerting on TRANCLASS

7.1 Creating a system event FileClose

This section describes how to implement a system event to generate an event if the file status is changed. It also can be helpful to allow you to start a backup batch procedure.

7.1.1 Creating zHFS directory

Use the TSO ISHELL to create a zFS directory (see Example 7-1) to export the bundle that contains the event binding file.

Example 7-1 Created zFS directory

```
/u/cicsrs1/EPRED_A01/SysEvent
```

7.1.2 Creating the bundle project

Complete the following steps to use IBM CICS Explorer to create the event binding file:

1. Open IBM CICS Explorer and select the resource perspective. Right-click in the Project Explorer view, click **File** → **New Wizards**, and click **CICS Bundle Project,** as shown in Figure 7-1.

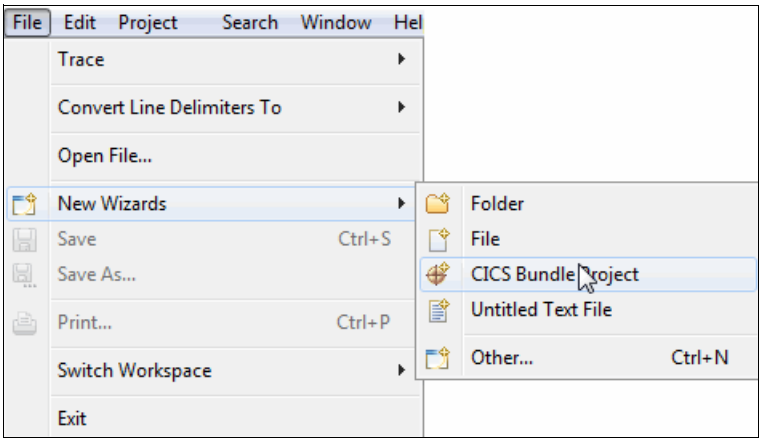


Figure 7-1 Creating CICS Bundle project

2. Specify the project name, `FileCloseEventBundle`, as shown in Figure 7-2 on page 133, and click **Finish**.

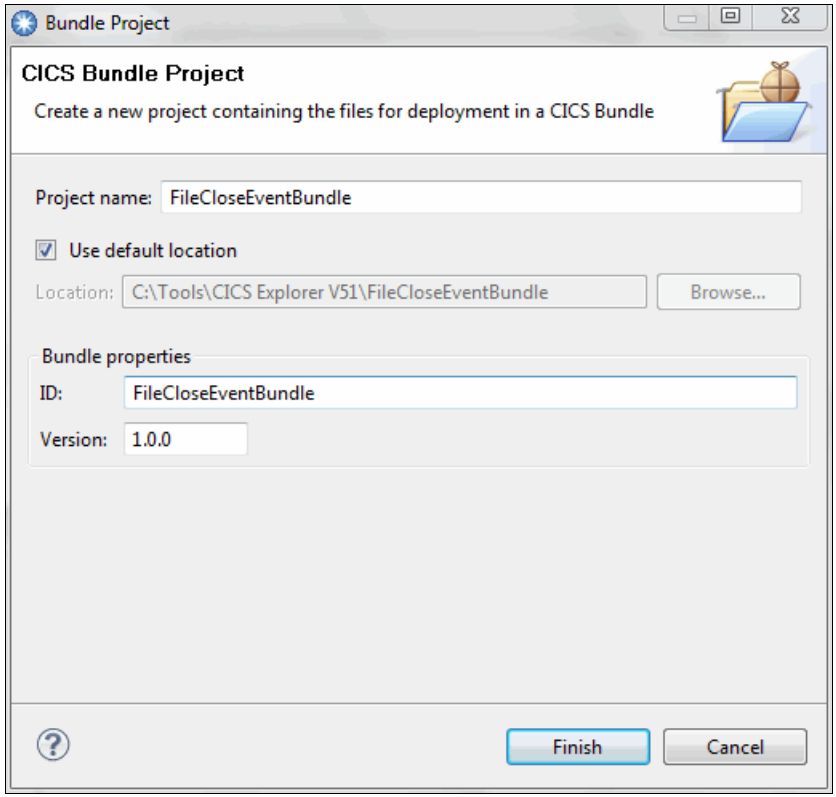


Figure 7-2 FileCloseEventBundle

3. Create the event binding within this bundle. In the Project Explorer view, right-click the project that was created, `FileCloseEventBundle`. Click **New** → **CICS Event Binding**, as shown in Figure 7-3.

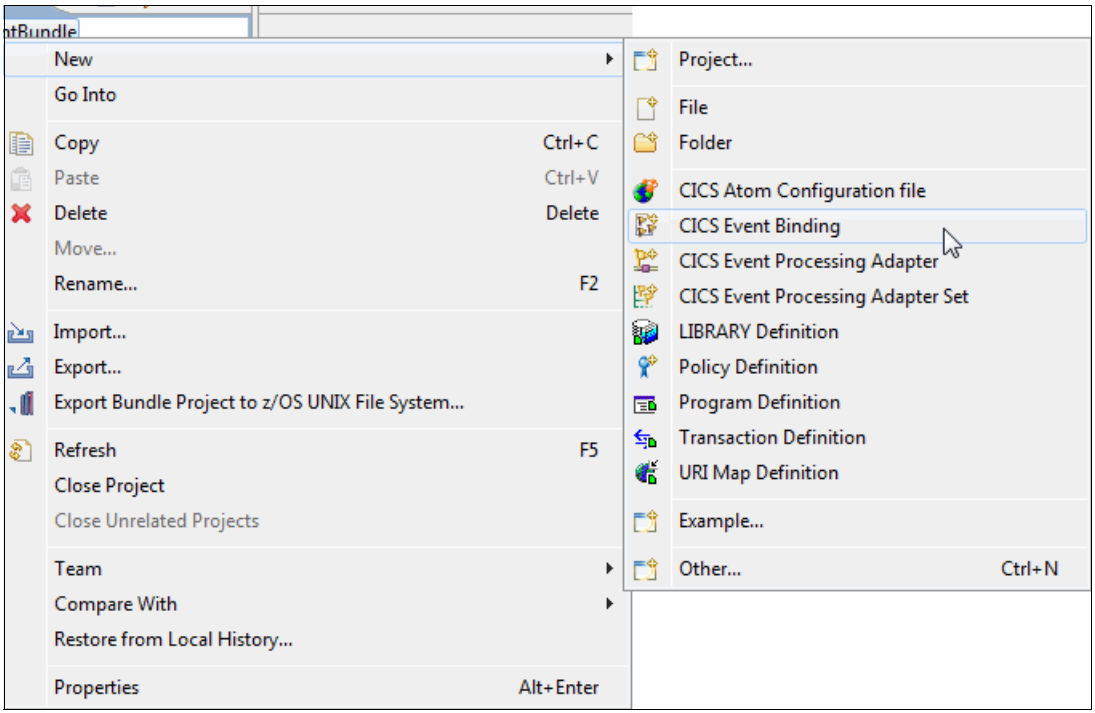


Figure 7-3 Creating Event Binding

4. Enter `FileCloseEventBinding` in the file name field and click **Finish**, as shown in Figure 7-4 on page 135.

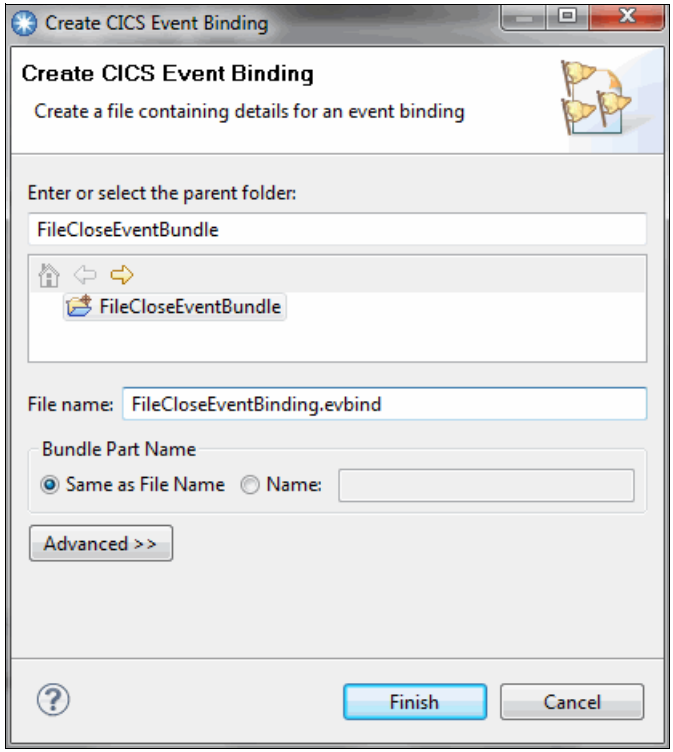


Figure 7-4 FileCloseEventBinding

Important: Figure 7-5 shows that some errors appear in the Problems pane, these are not concerns because they are caused by information not yet entered.

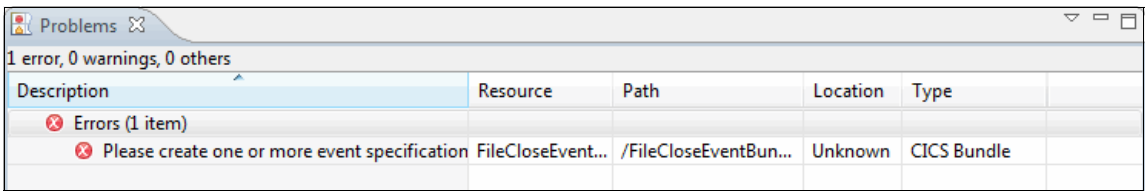


Figure 7-5 Creating event binding: Problems shown

5. Add an event specification to the event binding. In the view presented Figure 7-6 on page 136, enter a description for the event binding, and click **Add**.

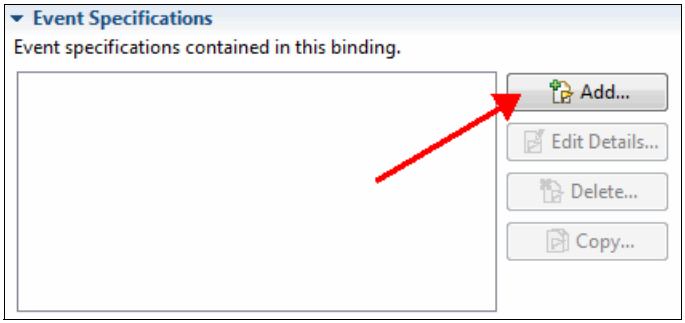


Figure 7-6 Add FileStockStatus binding specification

6. Enter the name of the Event Specification, *FileStockStatus*, a description, and click **OK** as shown in Figure 7-7.

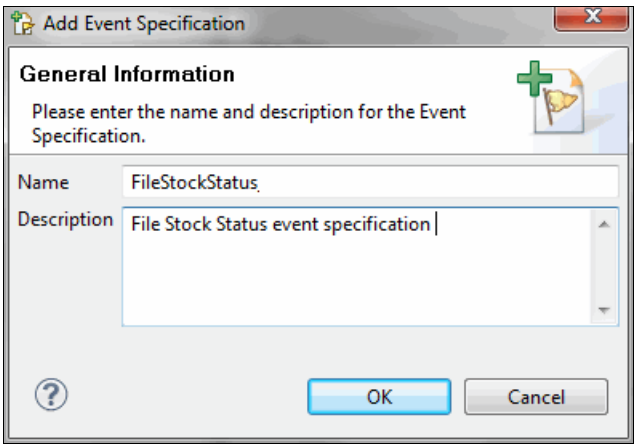


Figure 7-7 FileStockStatus event Specification

The system event will include the data set name, user ID and Task that closed the file STOCK, so add these as items of emitted business information.

7. You can now add the Specifications of the information you want to have when the system event is emitted. Click the Specification tab and then click **Add** (next to the Emitted Business Information table). In the Emitted Business Information view (as shown in Figure 7-8 on page 137), enter the following values:

   – Name: Dataset_Name
   – Type: Text
   – Length: Automatic
   – Precision: Automatic (leave as the default)

– Description: STOCK data set name

Click **OK**.

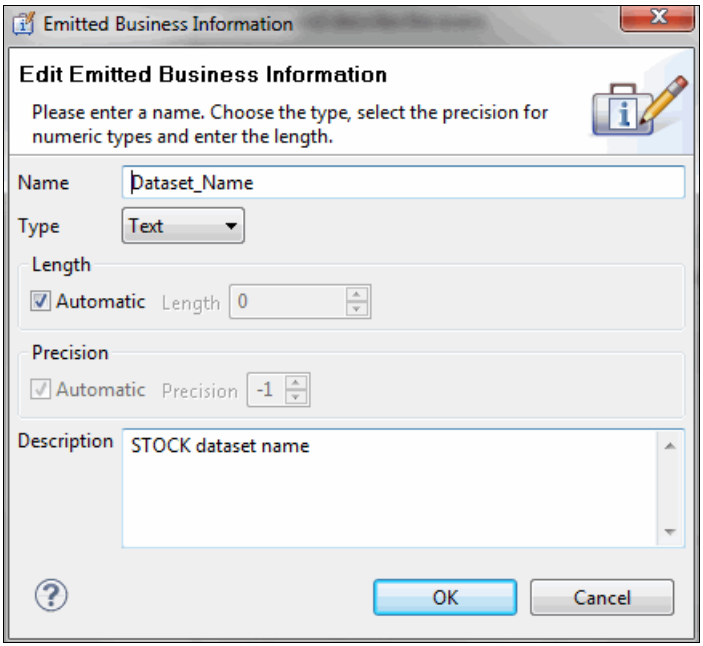


Figure 7-8 Add data set Name information

8. Repeat Step 7 to add the following user ID information:
   – Name: UserID
   – Type: Text
   – Length: Automatic
   – Precision: Automatic (leave as the default)
   – Description: User ID that closed the file

   Click **OK**.
9. Repeat Step 8 to add the following user ID information:
   – Name: TaskId
   – Type: Text
   – Length: Automatic
   – Precision: Automatic (leave as the default)
   – Description: TaskId that closed the file

   Click **OK**.

   The definitions are similar to those that are shown in Figure 7-9 on page 138.

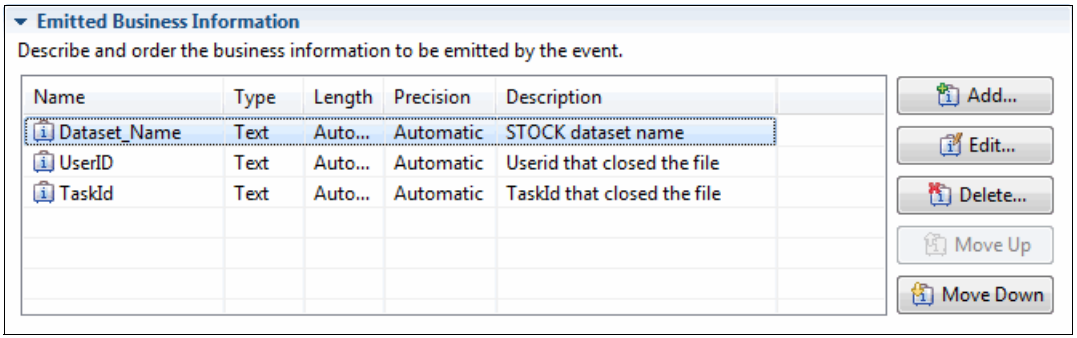


Figure 7-9 Emitted system events information summary

10. Add a Capture Specification to indicate the condition that trigger CICS to capture this event. In the Specifications view (see Figure 7-10), click **Add a Capture Specification**.

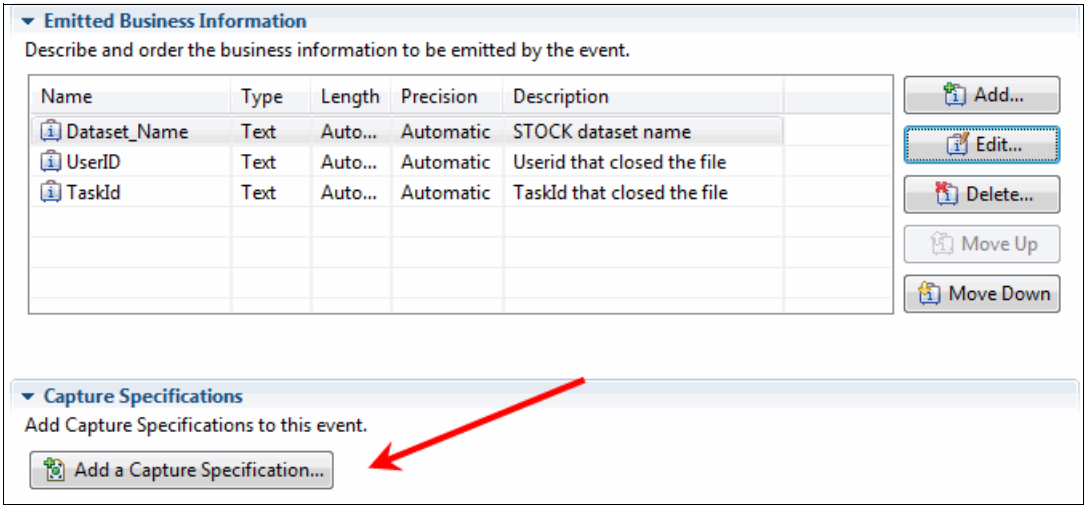


Figure 7-10 Event Specification view

11. In the Add Capture Specification view, enter the following information as shown in Figure 7-11 on page 139:
    - Name: FileCloseEventCapture
    - Description: Capture an event when STOCK is closed (see Figure 7-11 on page 139)

    Click **OK**.

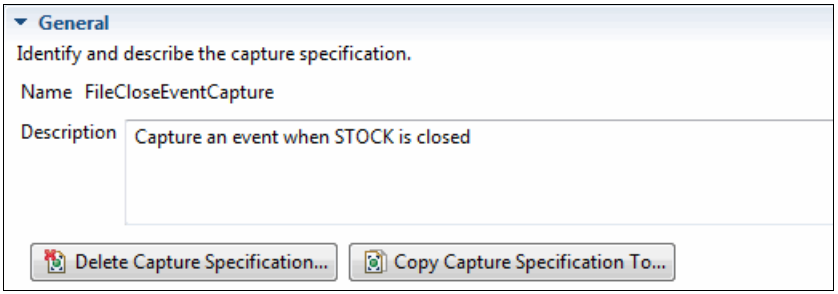


Figure 7-11 FileCloseEventCapture Capture Specification

12. In the next view, set the System Capture Point to FILE OPEN STATUS and click **Next:Filtering**, as shown in Figure 7-12.

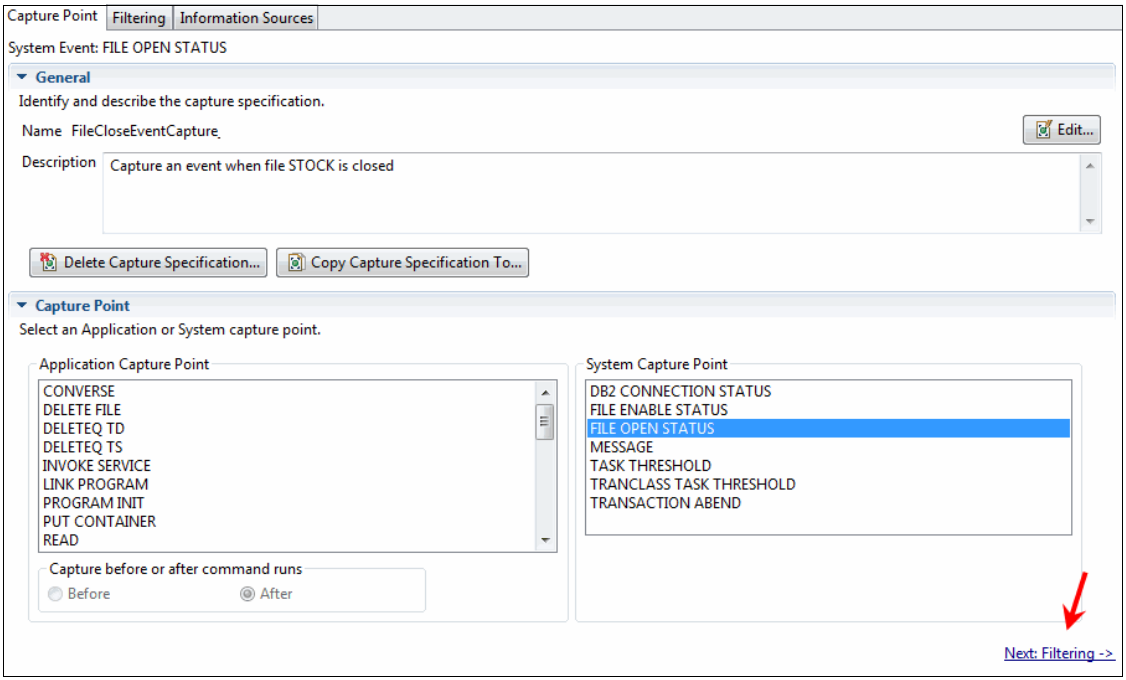


Figure 7-12 System event FILE OPEN STATUS capture point selection

13. The Filtering tab opens. Set the filtering for the event. Only events that match the following criteria are emitted:
    - In the File section:
      - Operator: Equals
      - Value: STOCK

– Select TO_OPENSTATUS radio button

  - Operator: Equals
  - Value: Closed

Enter the Filtering information as shown in Figure 7-13. In the Context part, leave all values at default.

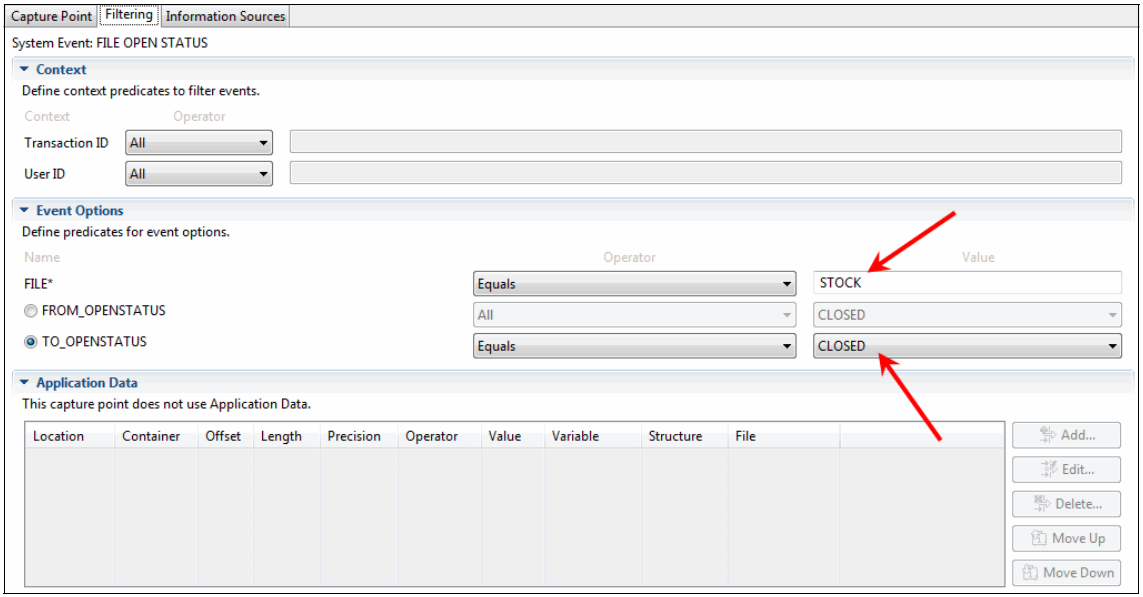


Figure 7-13 System Events filtering

14. Click **Next: Information Sources** on the Filtering panel to indicate how CICS can capture the requested data as emitted system information.

15. The Information Source tab opens. Because you already added Emitted Business Information (see Figure 7-10 on page 138), the Information Source panel (see Figure 7-14 on page 141) already is completed. You need only to match your source information with the existing one. If you do not see all the data items that you want included in the events, go back to the Emitted Information Items section in the event specification to add these items. To complete the information source, select the DataSetName information and click **Edit**.

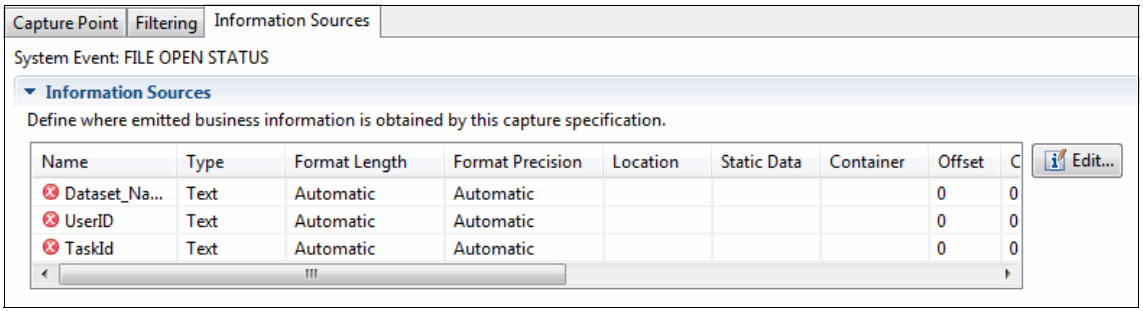


Figure 7-14 Information source specification

16. On the Edit Information Source panel, select DSNAME (that is, the data set name), as shown in Figure 7-15.

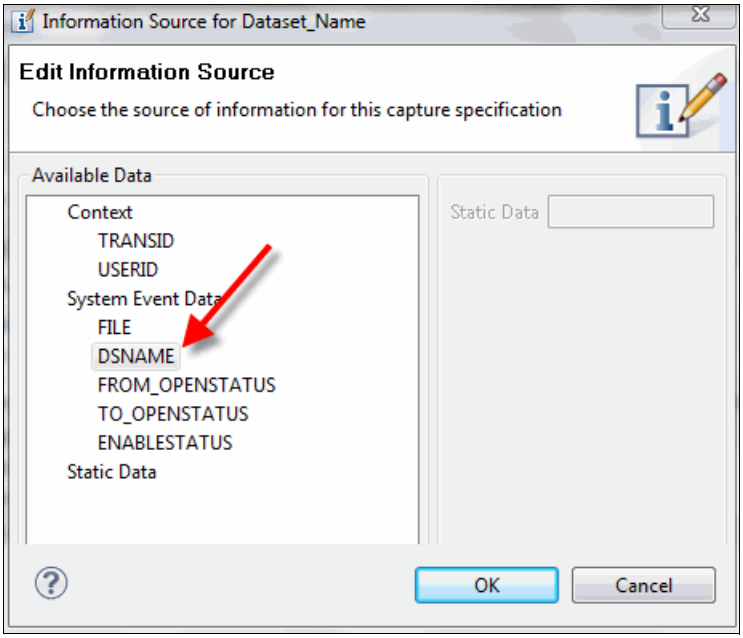


Figure 7-15 Select DSNAME information source

17. Repeat Step 16 for the user ID information, selecting **USERID**.
18. Repeat Step 17 for the TaskId information, selecting **TRANSID**.
19. Click **OK**. Figure 7-16 on page 142 shows a summary table of where the emitted business information is obtained.

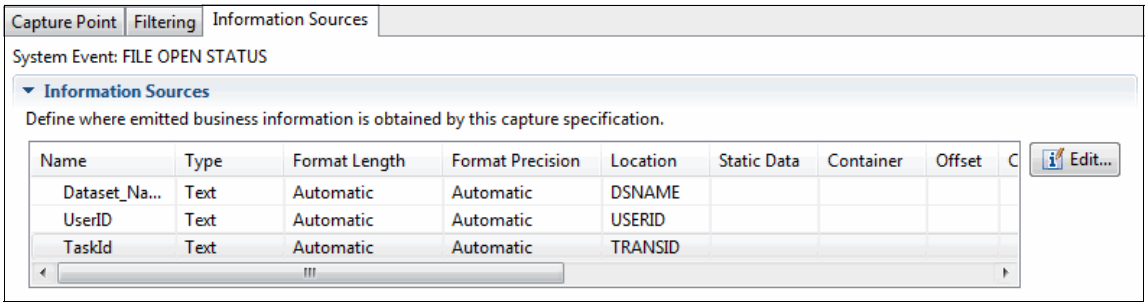


Figure 7-16 Information sources that are emitted

20. Click **Next: Adapter**.

7.1.3 Defining the Temporary Storage queue EP adapter

For test purposes, use the Temporary Storage Queue EP adapter. It allows you to easily verify the emitted system information by using the CEBR or the CECI CICS supplied transaction, or can even be used by a user transaction.

Also, you must consider that CICS allows a maximum of 32K ITEMs in a TSQueue, so in a production application of this sample, you must consider a clean-up procedure of the TSQ or to use another EP adapter, such as WebSphere MQ.

On the EP adapter panel (as shown in Figure 7-17 on page 143), select **Use an adapter defined here** and provide the following information:

- Adapter: TS Queue
- Queue: Name STOCKTSQ
- System: ID Leave blank (to use local TS queue)
- Use Auxiliary Temporary Storage: Leave cleared (use main storage TS queue)

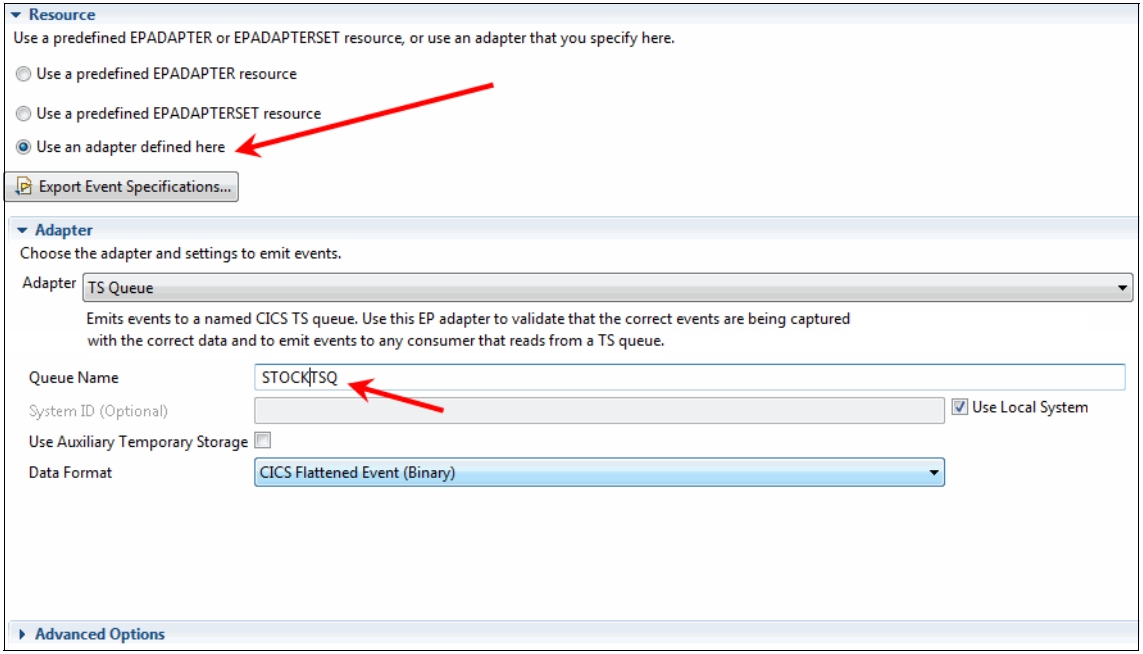


Figure 7-17 Use TSQ as EP adapter

Press Ctrl+S to save the configuration.

Important: An embedded EP adapter is used to simplify the example, but in practice a separate EP adapter gives more flexibility.

7.1.4 Exporting the bundle project

Export the bundle direct to a system zFS, where CICS can read it directly.

In the project view of IBM CICS Explorer, right-click **FileCloseEventBundle** and select **Export Bundle Project to z/OS UNIX File System**, as shown in Figure 7-18.

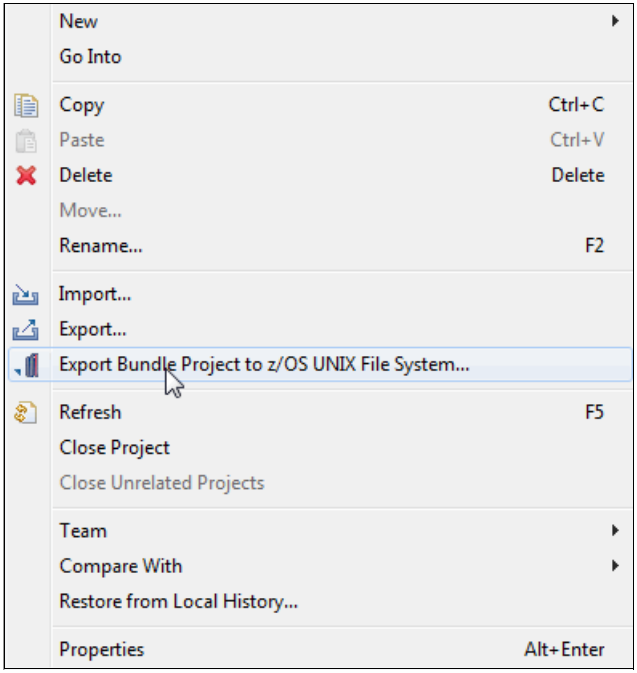


Figure 7-18 Export Bundle Project to zFS

Select to export in a specific directory, as shown in Figure 7-19.

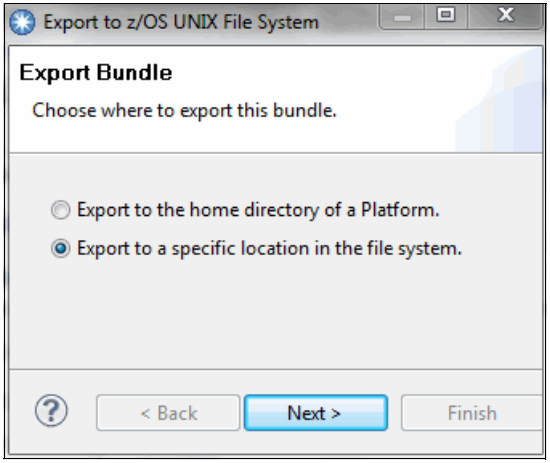


Figure 7-19 Select the directory for exporting

You now must select the zFS directory that contains your Bundle Project, as shown in the Figure 7-20 on page 146.

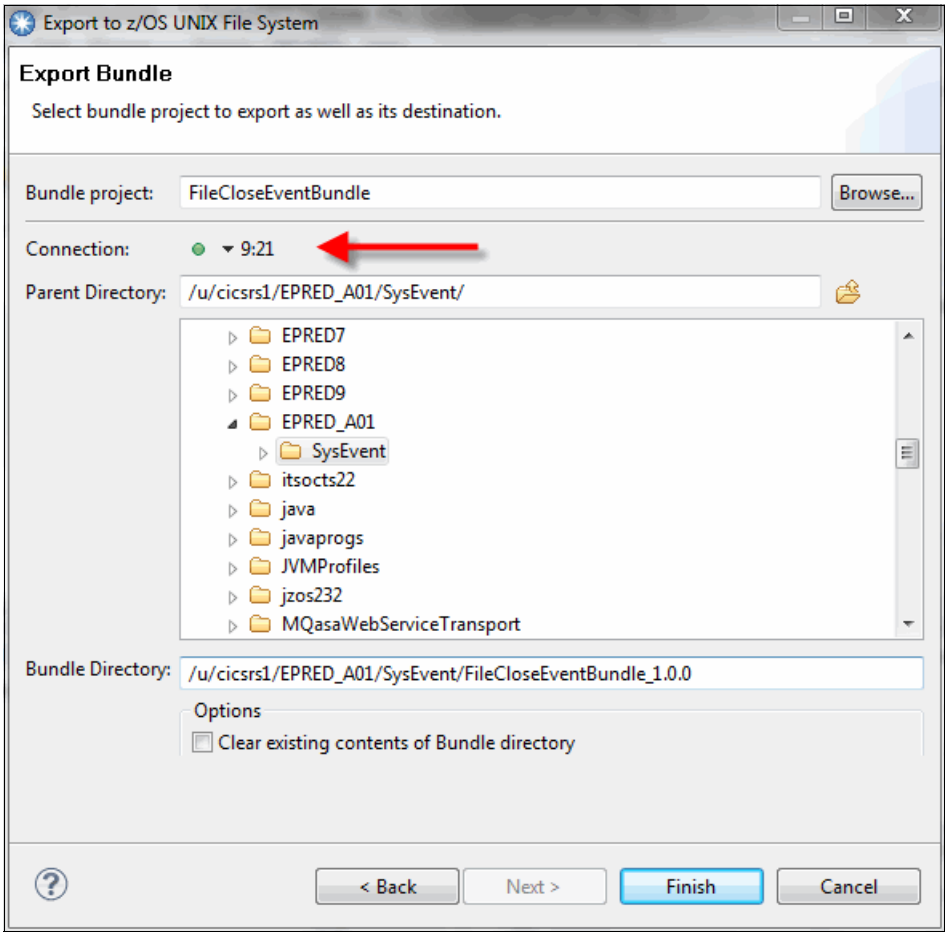


Figure 7-20 Export FileCloseEventBundle project

Important: You must be connected to z/OS FTP to export to system zFS the bundle project.

Important: It is advisable to select the Clear existing contents of Bundle directory option if the directory already exists to remove any pre-existing files.

After the export, we have the file structure that is shown in Example 7-2 on page 147 on zFS.

The export creates the following files:

- /u/cicsrs1/EPRED_A01/SysEvent/FileCloseEventBundle_1.0.0/META-INF/cics.xml
- /u/cicsrs1/EPRED_A01/SysEvent/FileCloseEventBundle_1.0.0/FileCloseEventBinding.evbind

Important: A directory name of FileCloseEventBinding_*version* is added by the export function. The name comes from the CICS Explorer project name and must be used in the CICS resource definition.

Version numbers allow you to have more versions of the bundle project.

Example 7-2 Exported bundle files

```
Dir /u/cicsrs1/EPRED_A01/SysEvent/FileCloseEventBundle_1.0.0/
   Dir   META-INF
      File     cics.xml
   File  FileCloseEventBinding.evbind
```

The exported CICS bundle manifest (cics.xml) is shown in Example 7-3.

Example 7-3 cics.xml file created by bundle export

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<manifest xmlns="http://www.ibm.com/xmlns/prod/cics/bundle"
build="BSF-20121101-1223" bundleRelease="0" bundleVersion="1"
bundleMicroVer="0" bundleMinorVer="0" bundleMajorVer="1"
id="FileCloseEventBundle">
    <meta_directives>
        <timestamp>2012-11-27T11:34:11.030+01:00</timestamp>
    </meta_directives>
    <define path="FileCloseEventBinding.evbind"
type="http://www.ibm.com/xmlns/prod/cics/bundle/EVENTBINDING"
name="FileCloseEventBinding"/>
</manifest>
```

The exported event binding file is shown in Example 7-4.

Example 7-4 FileCloseEventBinding.evbind file created by bundle export

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:eventBinding CICSEPSchemaVersion="2" CICSEPSchemaRelease="0"
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/cics/eventprocessing/
eventbinding CicsEventBinding.xsd "
```

```
xmlns:ns2="http://www.ibm.com/xmlns/prod/cics/eventprocessing/eventbind
ing" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <description></description>
    <userTag></userTag>
    <eventSpecification>
        <name>FileFileaStatus</name>
        <description>FileFileaStatus event specification</description>
        <eventInformation>
            <eventInformationItem name="Dataset_Name" dataType="text"
dataPrecision="-1" length="0" description="STOCK data set name "/>
            <eventInformationItem name="UserID" dataType="text"
dataPrecision="-1" length="0" description="Userid that closed the
file"/>
            <eventInformationItem name="TaskId" dataType="text"
dataPrecision="-1" length="0" description="TaskId that closed the
file"/>
        </eventInformation>
    </eventSpecification>
    <eventCaptureSpecification>
        <name>FileCloseEventCapture</name>
        <eventIdentifier>FileStockStatus</eventIdentifier>
        <description>Capture an event when STOCK is
closed</description>
        <filter>
            <contextFilter>
                <transactionId filterOperator="OFF" filterValue=""/>
                <currentProgram filterOperator="OFF" filterValue=""/>
                <userId filterOperator="OFF" filterValue=""/>
                <CommandResp filterOperator="OFF" filterValue="OK"/>
                <EIBAID filterOperator="OFF" value=""/>
                <EIBCPOSN filterOperator="OFF" filterValue="1"/>
            </contextFilter>
            <locationFilter filterType="SYSTEM">
                <fileOpenStatus capturePoint="FILE_OPEN_STATUS">
                    <FILE keyword="FILE" filterOperator="OFF"
filterFieldLength="8" filterValue=""/>
                    <FROM_OPENSTATUS keyword="FROM_OPENSTATUS"
filterOperator="OFF" filterFieldLength="16" filterValue="CLOSED"/>
                    <TO_OPENSTATUS keyword="TO_OPENSTATUS"
filterOperator="OFF" filterFieldLength="16" filterValue="CLOSED"/>
                </fileOpenStatus>
            </locationFilter>
            <dataFilter/>
        </filter>
        <dataCapture>
```

```
            <captureItem>
                <keywordCaptureItem source="DSNAME"
eventItemName="Dataset_Name" formatlength="0" formatdataType="text"/>
            </captureItem>
            <captureItem>
                <contextCaptureItem source="USERID"
eventItemName="UserID" formatlength="0" formatdataType="text"/>
            </captureItem>
            <captureItem>
                <contextCaptureItem source="TRANSID"
eventItemName="TaskId" formatlength="0" formatdataType="text"/>
            </captureItem>
        </dataCapture>
    </eventCaptureSpecification>
    <eventDispatcherSpecification>
        <eventDispatcher>
            <eventDispatcherPolicy>
                <dispatchPriority>normal</dispatchPriority>
                <eventsTransactional>false</eventsTransactional>
                <adapterUserid
useContextUserid="false"></adapterUserid>
                <adapterTranId></adapterTranId>
            </eventDispatcherPolicy>
            <eventDispatcherAdapter>
                <cicsTSQueueAdapter>
                    <queueName>STOCKTSQ</queueName>
                    <sysid></sysid>
                    <useAuxTempStorage>false</useAuxTempStorage>
                    <format>CFE</format>
                </cicsTSQueueAdapter>
            </eventDispatcherAdapter>
        </eventDispatcher>
    </eventDispatcherSpecification>
</ns2:eventBinding>
```

7.1.5 Installing the bundle in CICS

Use CICS Explorer to create the resources. In CICS Explorer, open the CICS SM perspective, as shown in Figure 7-21 on page 150.

In this example, use CICS Explorer to create and install the Bundle resources. However, you also can use CEDA, or other CICS Tools to define and install the resources that you need.

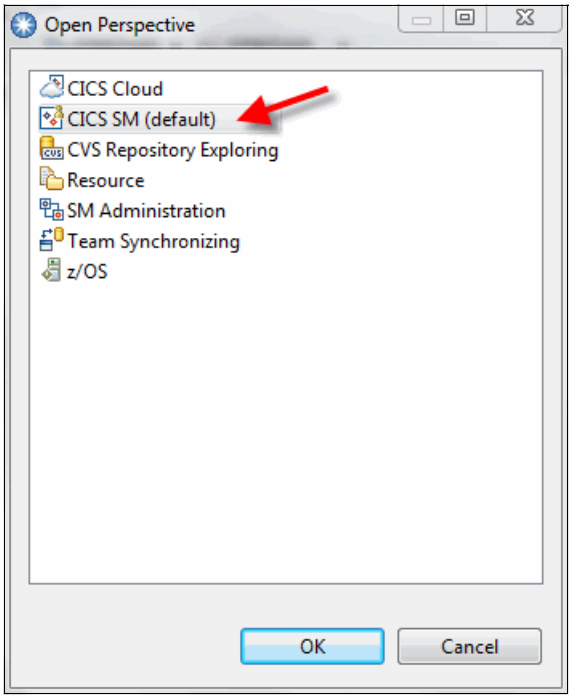


Figure 7-21 Open Perspective CICS SM

Complete the following steps to create the Bundle resource:

1. Click **Definition** → **Bundle Definitions**, as shown in Figure 7-22.

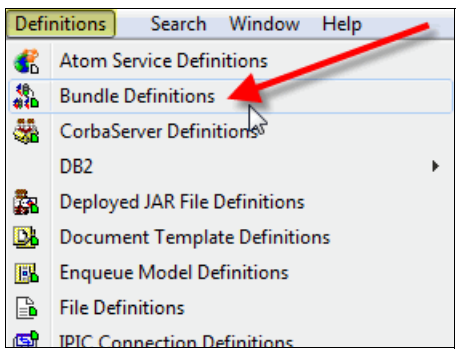


Figure 7-22 Bundle Definition to CICS

2. In the Bundle Definitions Window, right-click the white space.

3. Click **New** and enter the following information (as shown in Figure 7-23 on page 151):
   - Resource Group: EPSYSTEM
   - Name: EPSTOCK
   - Description: System Event on FILE STOCK
   - BundleDirectory:`/u/cicsrs1/EPRED_A01/SysEvent/FileCloseEventBundle_1.0.0/`

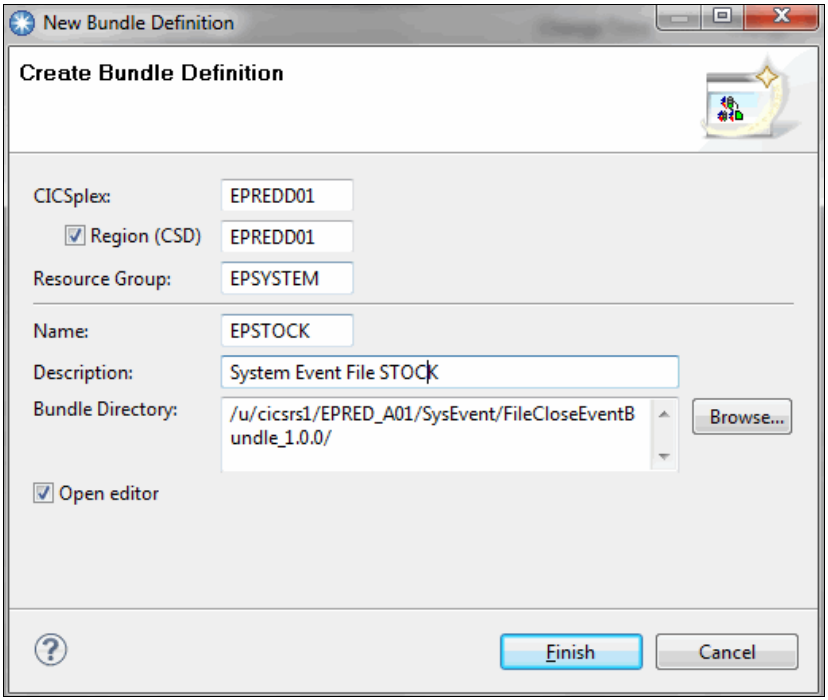


Figure 7-23 Create the CICS Bundle Definition

Click **Finish**.

Important: You must be connected to the z/OS perspective to browse the bundle directory.

The EPSTOCK Bundle Definition is shown on the Bundle Definitions window, as shown in Figure 7-24 on page 152.

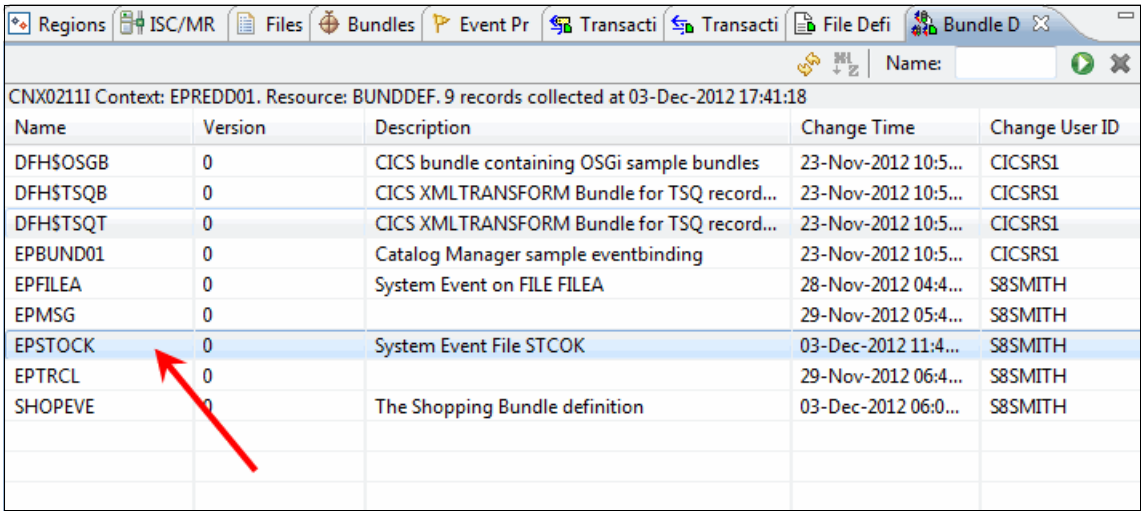


Figure 7-24 Bundle definitions window

4. In the Bundle Definitions view (see Figure 7-25) right-click the EPSTOCK bundle and click **Install.** In the CICSplex selection menu, you must select the CICS Applid where you want to install the Bundle. If you are connected to a single CICS region that uses SMSS connection, you see only the CICS where you are connected.

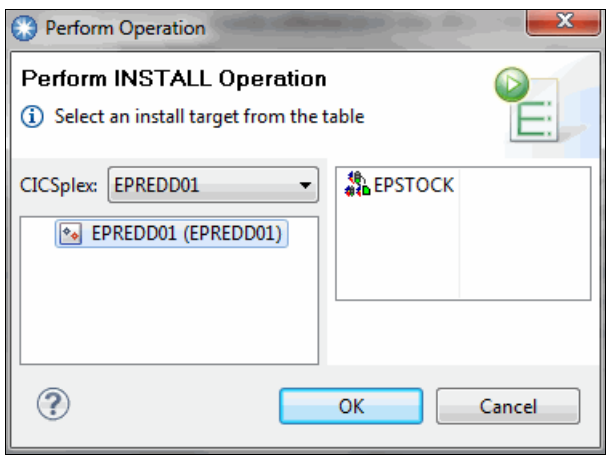


Figure 7-25 Install EPSTOCK Bundle definition on EPREDA01 CICS

5. Verify the install. In CICS Explorer, click **Operations** → **Bundles.** Figure 7-26 on page 153 shows the installed EPSTOCK bundle resource.

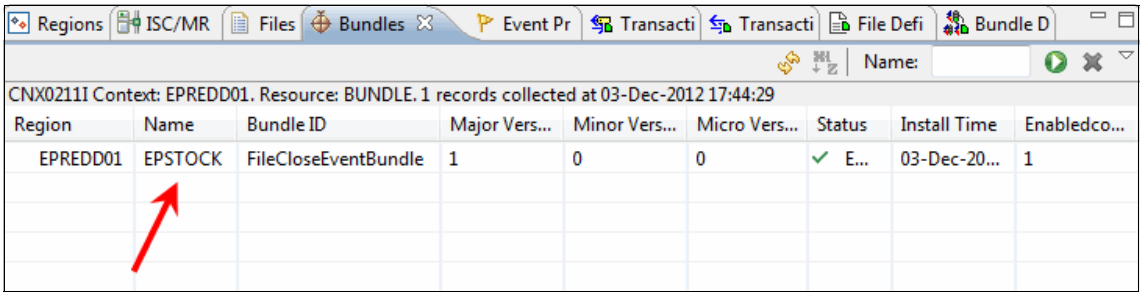


Figure 7-26 CICS Explorer bundle view of EPSTOCK resource installed

7.1.6 Testing the system event

Complete the following steps to test whether closing the `STOCK` file causes an EPSTOCK event to be emitted:

1. On a CICS SM perspective view, click **Operations** → **Files** → **Files** to check the file status (it must be Open and Enable), as shown in Figure 7-27.

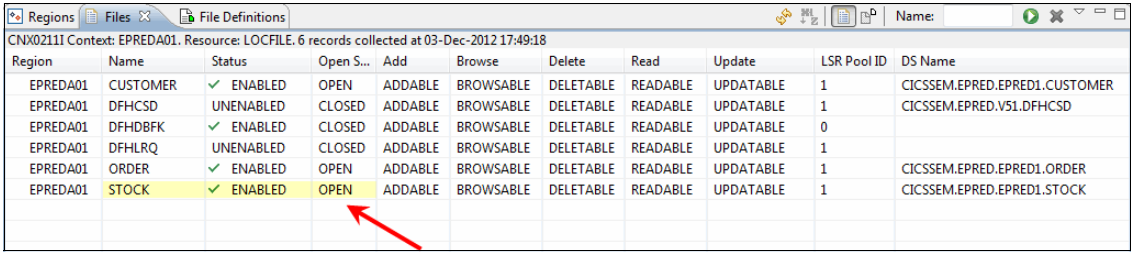


Figure 7-27 Check STOCK Status

2. Now you can close the `STOCK` file. Right-click the file and select **Close File** → **No Wait**, as shown in Figure 7-28.

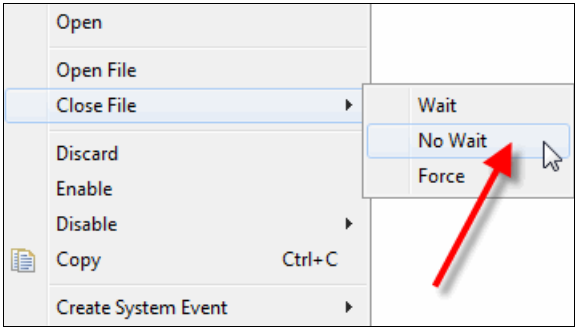


Figure 7-28 Close the STOCK file

7.1.7 Verifying the event processing

Complete the following steps to use CICSplex Explorer to verify event processing:

1. Select CICS region EPREDA01 within the CICS SM perspective. Click **Operations** → **Event Processing** → **Event Processing**, as shown in Figure 7-29).

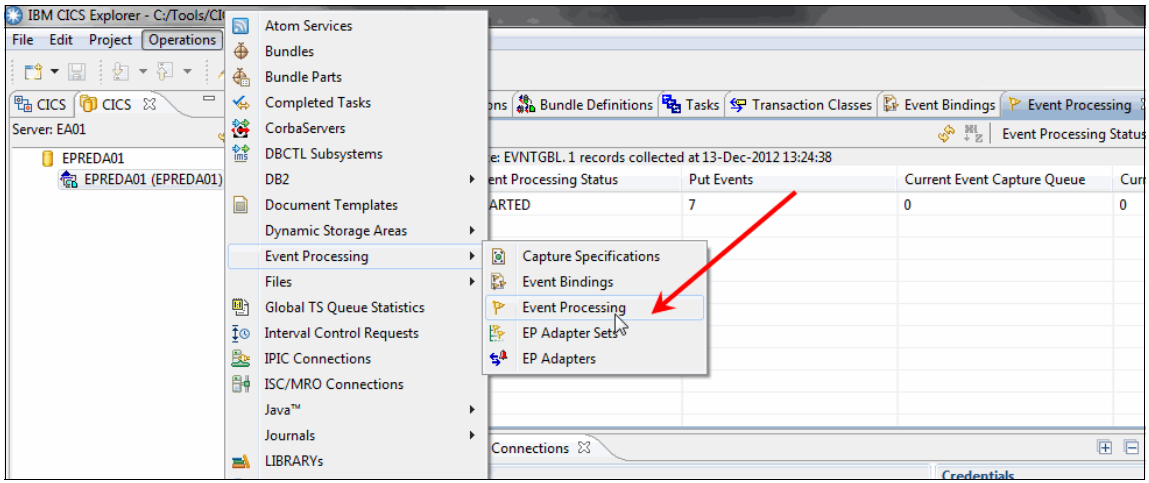


Figure 7-29 Check the status of Events Processing

Important: The column Put Event shows seven events because it identifies the number of put_events on the EP dispatcher queue. It is the sum of all the events that are emitted.

Column Put Events (see Figure 7-30) shows one event was emitted seven times.

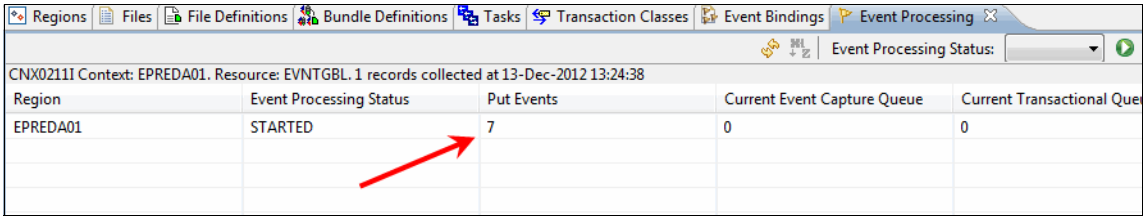


Figure 7-30 Events display

2. Verify that a record was written to the specified Temporary Storage queue STOCKTSQ. Use CICSplex Explorer to monitor the queue. The Item Count (see Figure 7-31) has a value of 1.

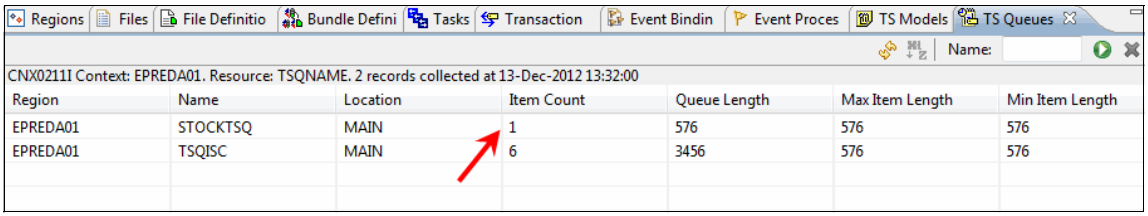


Figure 7-31 Temporary storage STOCKTSQ display

3. Log on to CICS and run the **CECI** command to read the temporary storage record that is written by the EP adapter, as shown in Example 7-5.

Example 7-5 CECI READQ TS command

```
CECI READQ TS QNAME(STOCKTSQ) ITEM(1) INTO(&DATA)
```

4. In the result panel, press PF5 to see variable. Place the cursor on the &DATA variable and press Enter to view the data, as shown in Figure 7-32.

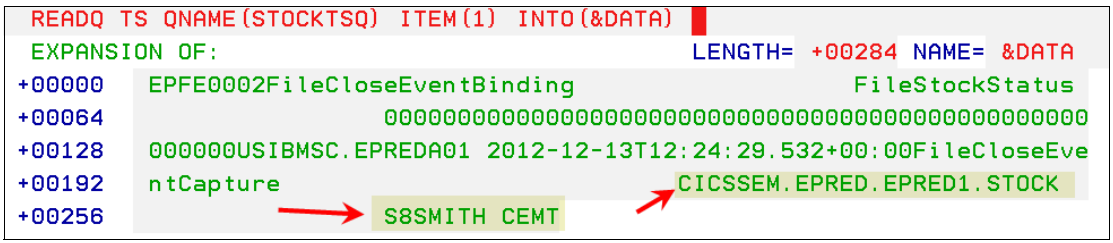


Figure 7-32 CECI READQ TS: Read Emitted event from STOCKTSQ

Important: The CICS-supplied transaction CEBR also can be used to display the queue, as shown in the following example:

```
CEBR STOCKTSQ
```

5. You can identify what you requested to be emitted: Data set name, Userid, and Transaction name that issued the close command to STOCK.

7.2 Sample system events for alerting on TRANCLASS

This section shows an example of the power of system events. Each step of the process that you use to run this test is described.

7.2.1 Background

A common situation that can happen in your system is that there is a task running in one CICS (that is, DOR/FOR) that compromises the productive running of other tasks in another CICS (that is, AOR).

For example, function shipping requests between CICS application-owning regions and connected file-owning regions can be queued in the issuing region while waiting for free sessions. Provided a file-owning region deals with requests in a responsive manner and outstanding requests are removed from the queue at an acceptable rate, then all is well. But, if a file-owning region is unresponsive, the queue can become so long and occupy so much storage that the performance of connected application-owning regions is severely impaired. Further, the impaired performance of the application-owning region can spread to other regions. This condition is sometimes referred to as *sympathy sickness*, although it should more accurately be described as intersystem queuing, which, if not controlled, can lead to performance degradation across more than one region.

It is valuable to identify this situation and respond to it rapidly, which avoids some outage exposure.

7.2.2 Goal

You can use system events if you have a critical application in your system and you want to be informed if the transactions are reaching the MAXACTIVE value and take actions to avoid application outage.

This sample helps you to send an appropriate message to the console alerting that TRANCLASS threshold is reached in your AOR. It also analyzes if the issue is because of a case of sympathy sickness for an unavailable DOR.

The sample emits an event if TRANCLASS TASK THRESHOLD is greater than 70% is reached in the AOR. The event starts a user transaction SSCK, as shown in Figure 7-33 on page 157.

SSCK performs the following tasks:

- Find the list of the suspended tasks in the TRANCLASS.
- If tasks are suspended in ZCIOWAIT, extract the UOWID of the tasks suspended in the TRANCLASS.
- Check if task is waiting on another CICS.
- Send a WTO to alert about TRANCLASS threshold that is reached. If the analysis drive to a specific CICS, inform about the possibility of Sympathy sickness on CICSDOR.
- System automation can now perform the required actions to reactivate the AOR-DOR connection, or route requests to another equivalent system (AOR-DOR pair).

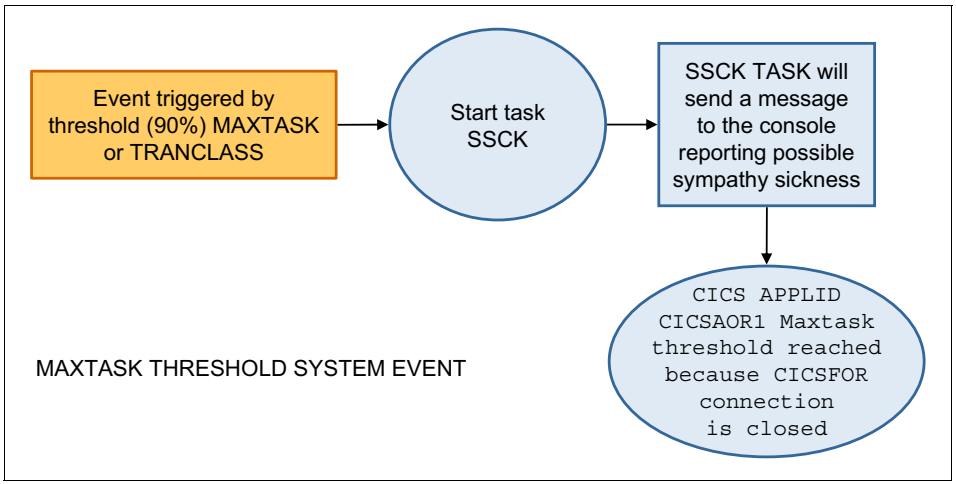


Figure 7-33 Sample description

This section does not describe each step because they are already described in 7.2.1, “Background” on page 156. Only the crucial steps are covered here.

7.2.3 System setup

We show setting up of connections only for the purposes of this example. It likely already exists in systems to which you might apply this approach.

Defining the connections

You must define an Advanced Program-to-Program Communication (APPC) connection between AOR and DOR. To do that, you must define Connection and session from AOR to DOR as shown in Figure 7-34 on page 158, Figure 7-35 on page 159, and Figure 7-36 on page 160.

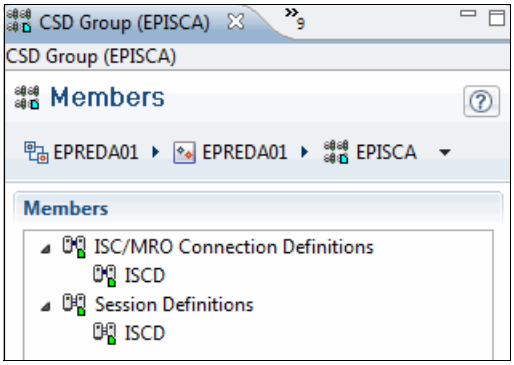


Figure 7-34 Define connection and session from AOR to DOR

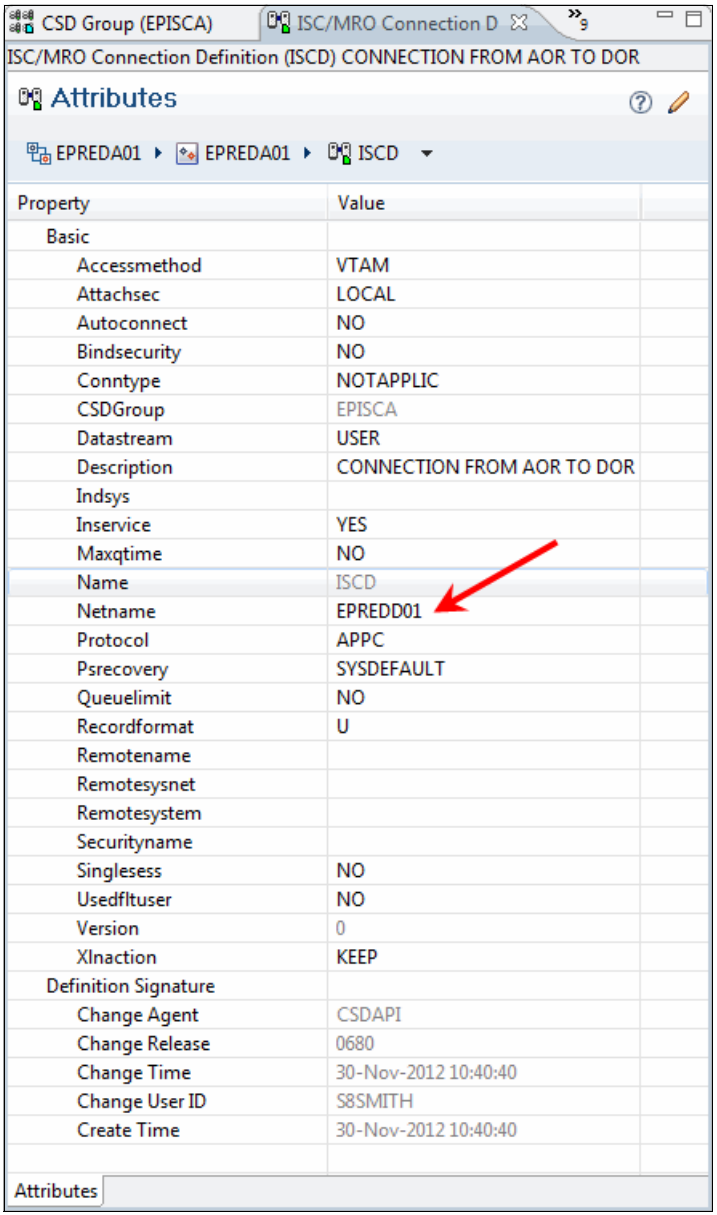


Figure 7-35 Connection Definition

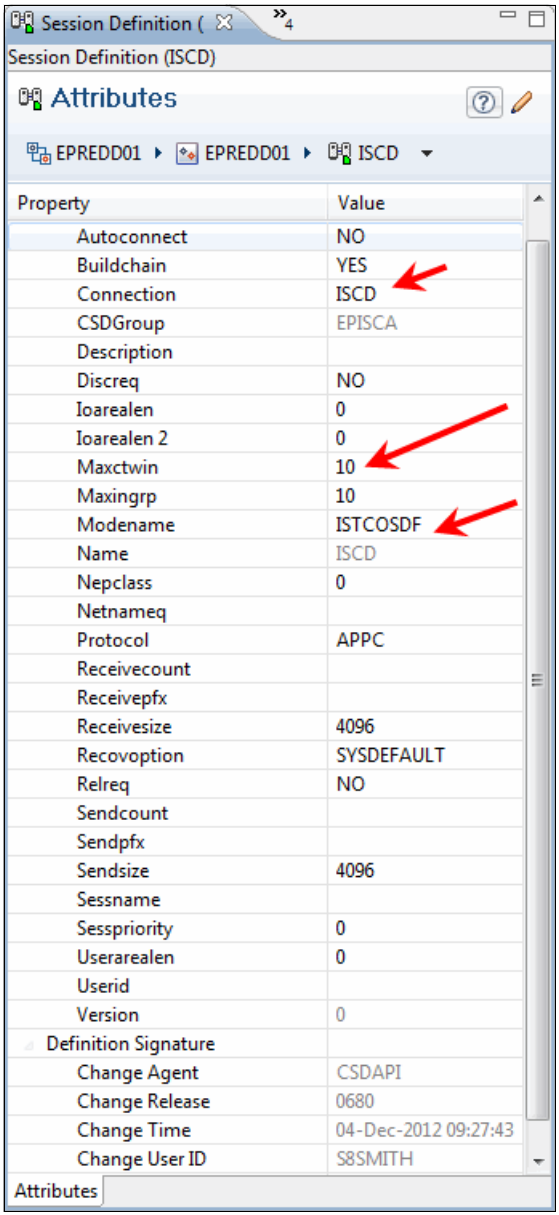


Figure 7-36 Session definition

You also must perform the same process for EPREDD01.

Defining the TRANCLASS

You must define a TRANCLASS to limit the number of contemporary tasks that are running on the system to test the provided sample.

Define a TRANCLASS like EPTRCL, shown in Figure 7-37.

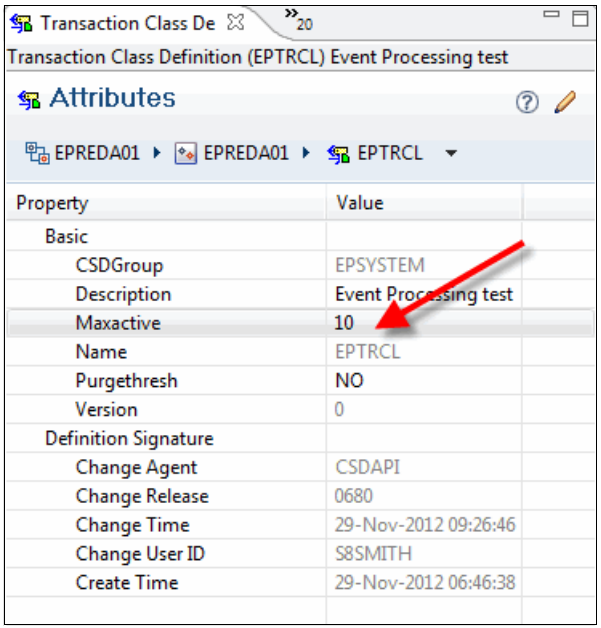


Figure 7-37 Transaction Class definition

7.2.4 Application sample setup

This section describes how to install all the resources that are needed to run the sample application

AOR definitions

Complete the following steps to define the AOR:

1. Run the **`CEDA COPY ALL GR(EPRED1) TO(EPAFLA)`** command.
2. Enter the group EPAFLA in your Resource group in your explorer, as shown in Figure 7-38 on page 162.

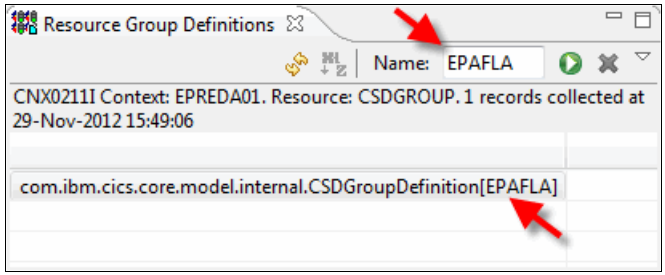


Figure 7-38 Display the resource group EPAFLA

Double-click the Resource group.

3. CSD group EPAFLA is displayed as shown in Figure 7-39 on page 163.

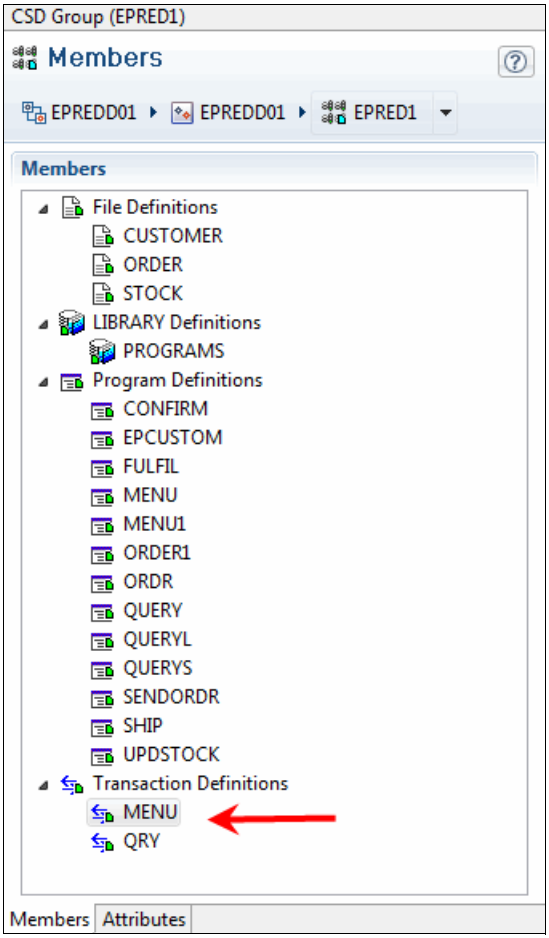


Figure 7-39 EPAFLA group display

4. Modify all the transaction definitions to have the EPTRCL TRANCLASS attribute as shown in the sample in Figure 7-40 on page 164.

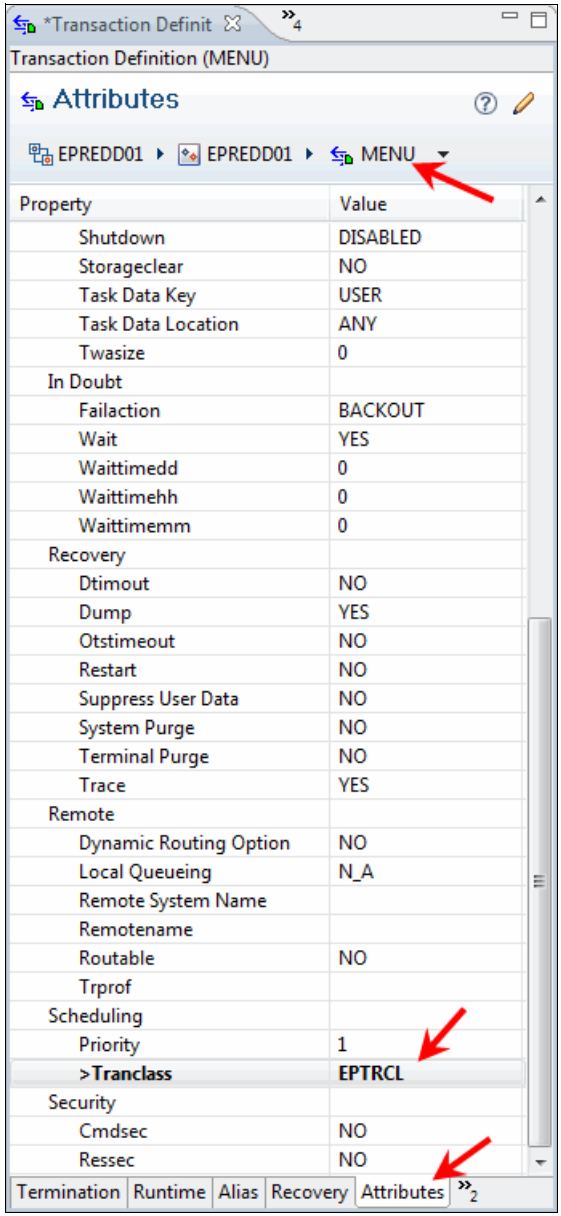


Figure 7-40 Modify the attribute of MENU transaction definition

5. Press Enter to confirm and Ctrl+S to save.

Defining the file

Complete the following steps to define the file remote to DOR:

1. Modify STOCK, ORDER, and CUSTOMER files as remote.

   Click the file and modify the remote attribute, as shown in Figure 7-41.

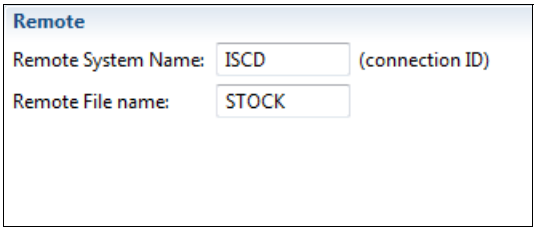


Figure 7-41 STOCK remote definition in AOR

2. Add the group EPAORFIL to the LIST EPAORLST, as in Figure 7-42.

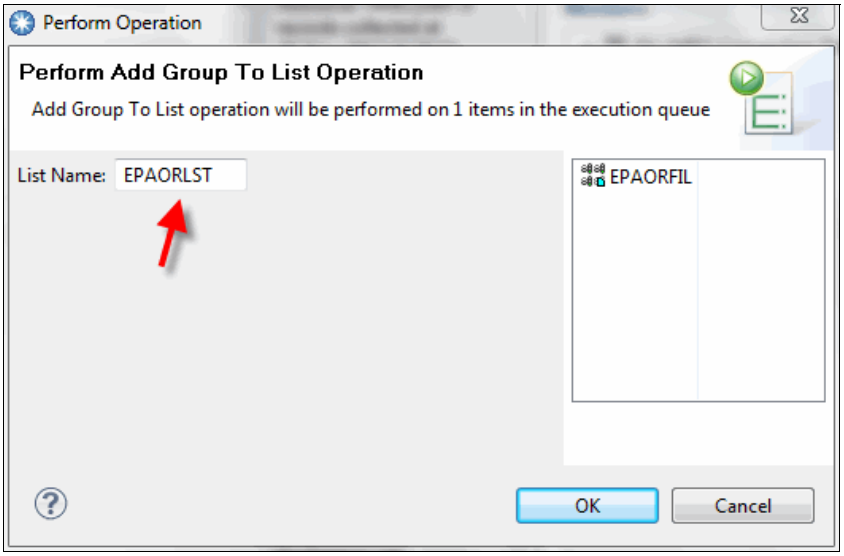


Figure 7-42 Add groups to list

DOR definitions

You can use the Group EPRED1. You only must add it to a group and append to the EPDORLST.

7.2.5 Defining an event on TRANCLASS TASK THRESHOLD

This event is emitted if TRANCLASS TASK THRESHOLD greater than 70% is reached.

After you define Bundle project and bundle event binding, you can define the capture point as shown in Figure 7-43.

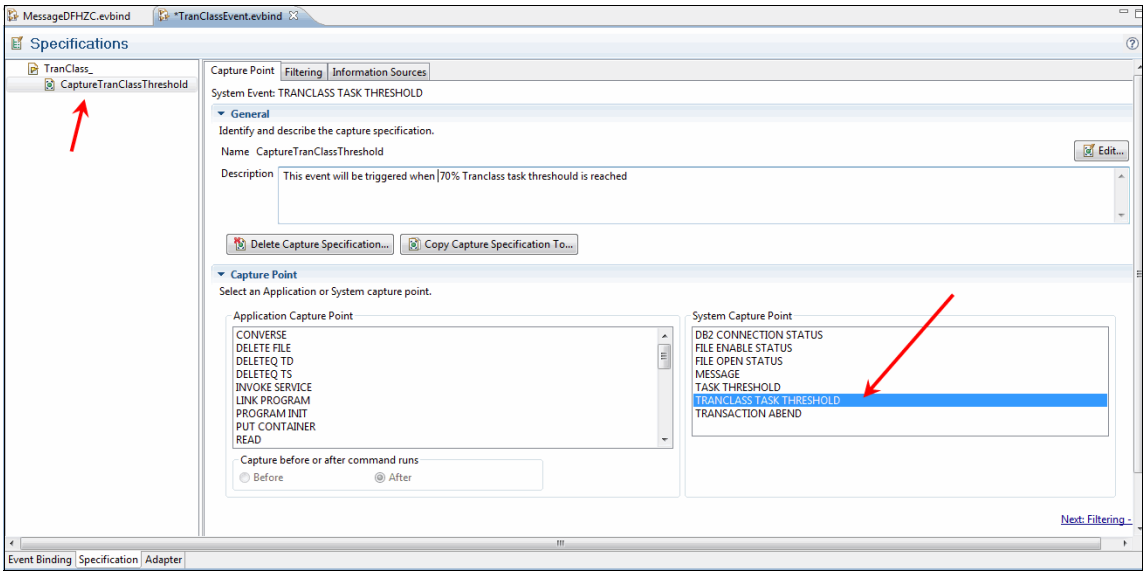


Figure 7-43 Capture point for Tranclass task threshold

In this example, we are filtering for a 70% threshold on tranclass EPTRCL shown in Figure 7-44.

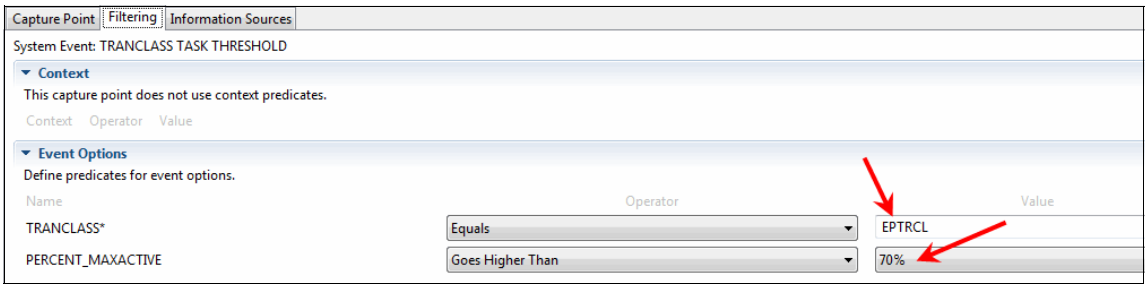


Figure 7-44 Filters for Tranclass task threshold

3. The source information contains the fields that are as shown in Figure 7-45.

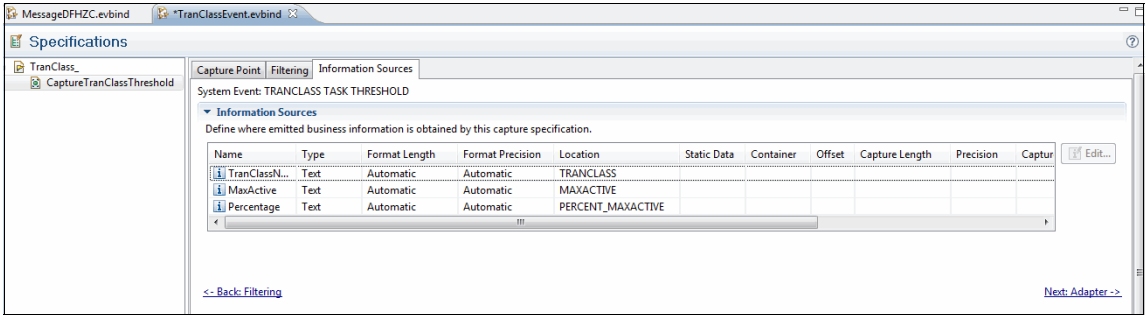


Figure 7-45 Information source for Tranclass task threshold

You can now define the EP adapter. It starts the user transaction SSCK**,** as shown in Figure 7-46.

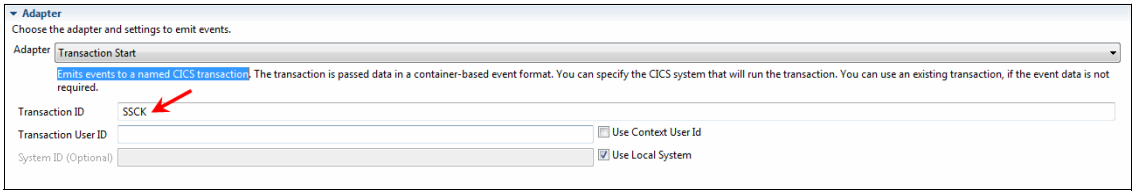


Figure 7-46 EP adapter used by Tranclass task threshold event

7.2.6 Test Case

To reproduce the condition, run the MENU transaction under CEDF in DOR until the START BROWSE locks the file ORDER.

You must select F4 - Fullfill order to go in the start browse.

Now run MENU on AOR seven times. The tasks hang because the file is locked by the start browse.

7.2.7 User Task SSCK

The example shows an easy application that identifies a simple logic if the TRANCLASS threshold reached is because of a sympathy sickness condition.

This program is only a sample and it does not cover all of the possible situations that can drive to a sympathy sickness condition.

The techniques and the code that are used show only the possibility that is provided by system events.

The first thing that the task SSCK does is provide an inquiry of the task status, as shown in Figure 7-47.

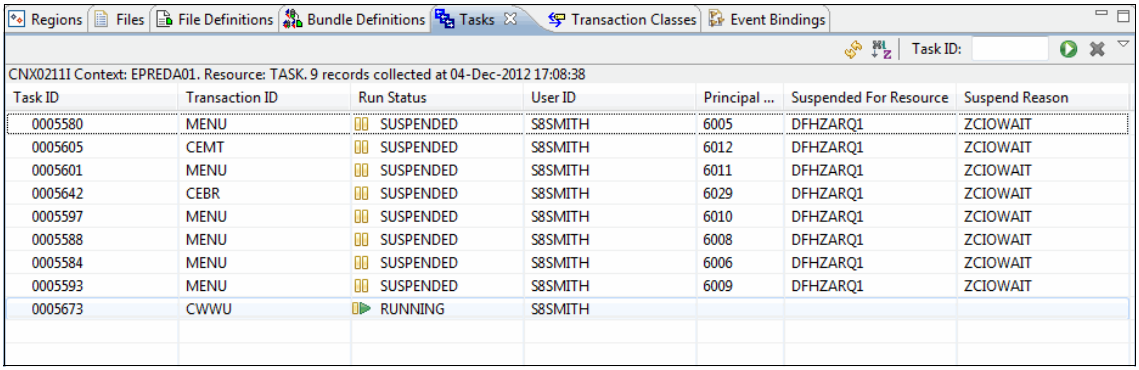


Figure 7-47 CICS Explorer Operation: TASKS

You can see that all are in ZCIOWAIT. Now we must understand why and where.

The information is available only by using the **`INQUIRE UOWLINK`** command.

At this point, we can guess that there is something that should be checked in EPREDD01 to avoid the TRANCLASS MAXACTIVE condition. A WTO is sent to the console and the system automation performs the action that is required to recover the situation.

The source of the program can be found in the Appendix B, “Additional material” on page 243.

You must install the definition of task SSCK and program SSCKPGM.

Part 3

Working with CICS events

This part of the book focuses on other IBM products such as, IBM Operational Decision Manager and IBM Business Monitor being integrated into the event processing in CICS and provides step-by-step instructions.

This part starts by providing a method to gauge the effect that changes to an application might have on the events in a CICS system. It then describes best practices for performance considerations. Problem determination and troubleshooting also are discussed.

8

Governance and troubleshooting

This chapter provides a method to gauge the effect that changes to an application might have on the events in a CICS system. It also describes best practices for performance considerations and event processing monitoring and statistics. The problem determination section shows a procedure to determine why expected events are not captured, and gives guidance to ascertain why too many events are captured and why, after data has been captured, it might not be as expected.

This chapter includes the following topics:

- Impact of application changes on events
- Best practices for performance
- Monitoring and statistics
- Problem determination

8.1 Impact of application changes on events

To assess the effect changes to applications might have on events in a CICS system, a clear understanding is needed of the resources that are named in event bindings that are currently installed in the system. Equally as important is identifying resources named in event bindings that are currently defined in the CICS Explorer workspace and might be installed in a CICS system. A new search function that is specifically designed for event processing artifacts can help with this process.

8.1.1 Event processing search

The event processing search function, which is initiated from the CICS Explorer Resource perspective, allows for the analysis of installed and offline resources that are used in the capture and emission of events. To start a search, select the Search function from the menu bar, as shown in Figure 8-1.

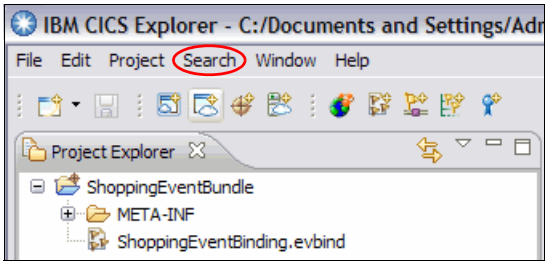


Figure 8-1 Selecting the event processing search function in CICS Explorer

An offline search can be initiated against different resource types and information sources within the event bindings in the CICS Explorer workspace by specifying a search string and selecting or clearing the Case sensitive option.

The search can be limited by selecting the options in the EP Search panel, to search for a specific resource type that can appear in the context data or the event options of an event binding, for example.

In the Shopping example that was used in earlier chapters of this book, a search can be run to find a resource type of PROGRAM in context data. Selecting the box labeled **Ignore predicates with operator "All"** limits the search results that are returned to only those options that contain other operators, as shown in Figure 8-2 on page 173.

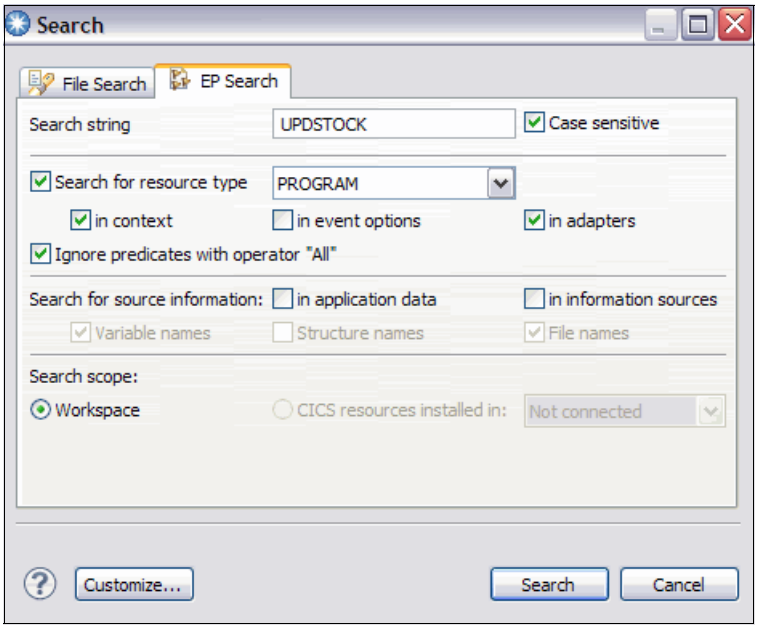


Figure 8-2 Search the current workspace for a PROGRAM resource called UPDSTOCK

The search results return all exact and potential hits. For example, if an event binding included a current Program with a name of PROG in the context data and the operator used is “Does not start with”, this event binding is included in search results for a program name of UPDSTOCK. Ensuring that the **Ignore predicates with operator “All”** is selected limits the number of potential search results. This configuration might risk missing potential effects. For example, if an event specification captures WRITE FILE requests to a file CUST from programs with any name, the event specification uses the All operator for the current program and a search for program UPDSTOCK only finds such an event specification if the Ignore predicates with operator All is not selected. In this situation, a search for file CUST might be a more appropriate way to find the impact of changes.

In this instance, the search results that are only limited to the workspace include the Capture fulfill capture specification in the `ShoppingEventBinding` file. Switch to the Search Results view in CICS Explorer, which is displayed when the search is started. Click **Expand All** to see the results of the search, as shown in Figure 8-3 on page 174.

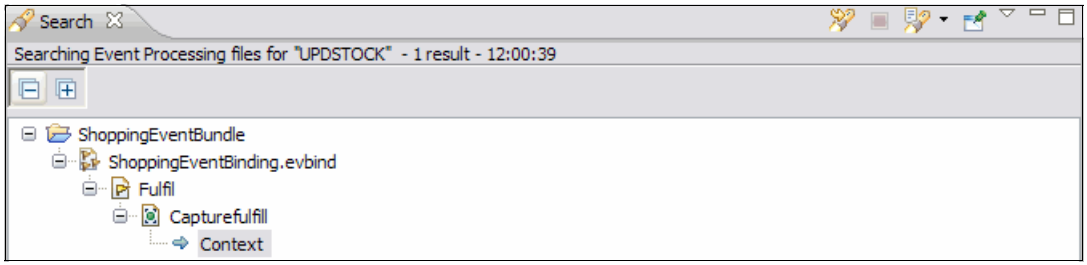


Figure 8-3 Workspace search results for the UPDSTOCK program resource

Double-click the word **Context** in the search results view to open the ShoppingEventBinding in the Event Binding Editor. The capture specification filtering page is displayed, where the program name UPDSTOCK can be found in the Context section.

To search all installed event bindings, use CICS Explorer to set a System Management Host connection to a CICSplex (see 5.1.2, “CICS Explorer connectivity” on page 83). Change the scope in the EP Search panel to CICS resources installed in. Select the CICSplex from the available list, as shown in Figure 8-4.

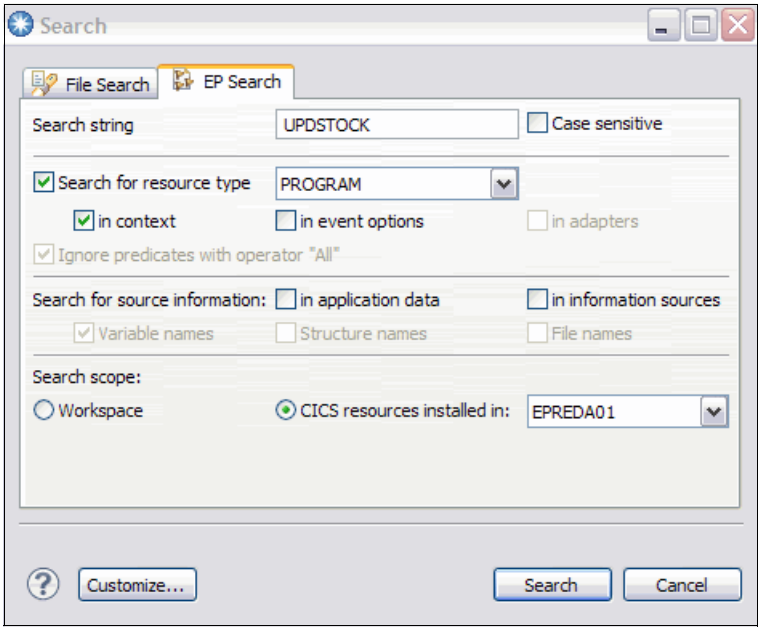


Figure 8-4 Search a CICSplex for a program resource called UPDSTOCK

Switch to the Search Results view in CICS Explorer and expand the output. The results are returned for each CICS region in the CICSplex, as shown in Figure 8-5.

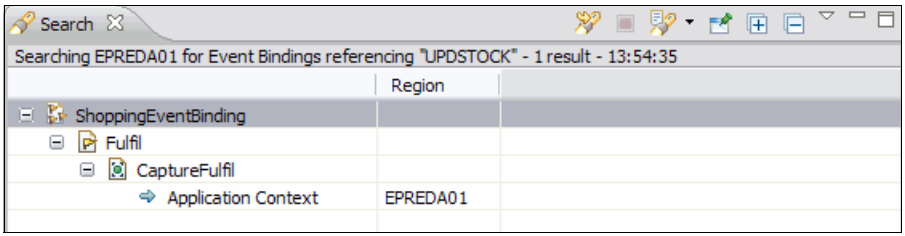


Figure 8-5 Search results for installed event bindings in a CICSplex

Double-click **Application Context** or right-click **Application Context** and select **Open** to view the Capture Specification for the ShoppingEventBinding event binding. The value UPDSTOCK can be seen in the current program (`Currpgm`) field, as shown in Figure 8-6.

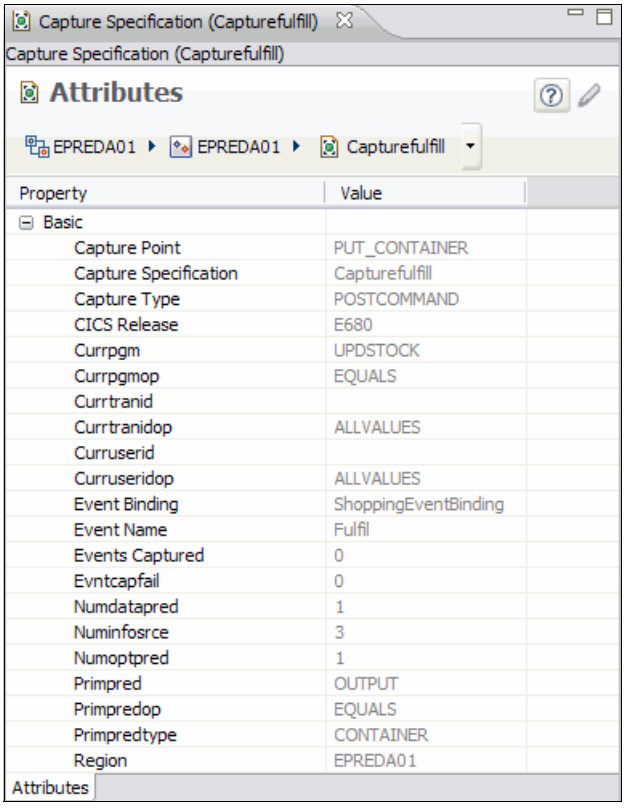


Figure 8-6 Capture specification details for an event binding installed in a CICSplex

8.2 Best practices for performance

From a performance perspective, CICS event processing consists of the following main steps:

- Capturing an event
- Dispatching the EP adapter
- Emitting the event through the EP adapter

8.2.1 Capturing an event

Filtering of the predicates that are specified for each event occurs during the capture stage. The capture point specifies the API call or system occurrence for which an event is to be captured. The filter is made up of predicates that identify the circumstances in which instances of the API call or system occurrence relate to the event. So, when event processing is started and no event bindings are installed, processor usage is negligible. Processor usage is also negligible when event bindings are installed, but there is no capture specification that matches the capture points that are started in the system.

The primary predicate is the primary value that is specified in the API command or on the system occurrence. For example, for a **`PUT CONTAINER`** command, the container name is a primary predicate. On a READ FILE and on a FILE ENABLESTATUS or OPENSTATUS capture point, the file name is a primary predicate. On a Filter, performance is optimized by the use of primary predicate filters that use the Equals operator. If there is a match on the primary predicate, use of Equals can result in improved performance when compared to other operator values.

Reduce the number of filters, where possible, by not including unnecessary filters. For example, if only one program writes to file XYZ, it is necessary to filter only on the file name and not also the program name.

Another way to help with performance is to organize the filter predicates to put those that filter out the largest number of unwanted events before the filter predicates that exclude fewer unwanted events in the capture specification.

Do not use filter predicates on zoned decimal or packed decimal data fields that might contain unusable data. Ensure you have sufficient filter predicates before making a packed decimal or zoned decimal check to ensure that the data area that is checked always contains valid packed decimal or zoned decimal data.

If you capture payload data that is associated with the event (information sources in the CICS Explorer event binding editor), keeping the number of data items captured to a minimum aids performance. For example, it is more efficient to define a single data capture element to capture bytes 0 - 35 than it is to define five data capture elements capturing bytes 0 - 5, 10 - 15, 20 - 25 and 30 - 35. It is recommended that containers do not contain a mix of data types. If the event is emitted to another platform, it is important that character data is translated to the correct code page.

8.2.2 Dispatching the EP adapter

In the dispatching stage, event processing runs multiple event dispatcher tasks on L8 TCBs to emit events. The maximum number of L8 TCBs is fixed at two times the value of MXT, plus 32. As of CICS TS V5.1, the maximum number for event processing dispatchers is limited to one third of this value to prevent event processing from monopolizing the L8 TCBs. Use the data in the “Peak event capture queue” and “Peak dispatcher tasks” entries in the EVENTPROCESS global statistics to ensure that the max tasks value is not set too low.

Specifying a user ID or transaction ID in the advanced options of the EP adapter causes the EP adapter to be attached as a separate task. However, performance is improved if no user ID or transaction ID is specified in the EP adapter because the EP adapter is then linked to the dispatcher task.

8.2.3 Emitting an event through the EP adapter

During the EP adapter stage, the cost of running the EP adapters depends on which EP adapter is used. The TSQ EP adapter imposes the least cost in terms of CPU usage.

An MQPUT1 call is generated for each event that is emitted to the WebSphere MQ EP adapter, which has some overhead. In addition, if event emission is assured (synchronous emission option), the cost of this call transfers from an asynchronous event processing task to the application thread, which can increase application response time.

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

The HTTP EP adapter issues a WEB OPEN, WEB CONVERSE, and WEB CLOSE call for each event and by default closes the client HTTP connection after the event is emitted. Processor overhead for reopening the connection can be saved by keeping the connection open after event emission. You can achieve this by specifying the SOCKETCLOSE attribute in the URIMAP resource that the HTTP EP adapter names. A CEPH task is attached for each event that is emitted by using the HTTP EP adapter, which has a default timeout value (RTIMEOUT) of 5 seconds. By taking a copy of the profile for this transaction (DFHECEPH) and the CEPH transaction, the timeout value can be changed to match your environment. For example, if the event emission rate is high when compared to the response rate of your network or HTTP 1.1 compliant server, the number of CEPH tasks that are generated can cause the MXT limit to be reached on your system. To avoid this, assign a copy of the CEPH transaction to a transaction class that has a MAXACTIVE value low enough to avoid reaching the MXT limit.

For the same number of event emissions, the use of an EP adapter set, separate EP adapters, or EP adapter that is defined as part of an event binding all have similar performance characteristics.

8.2.4 Assured events

The following scenarios cause an increase in CPU usage:

- When a WebSphere MQ EP adapter is used, synchronous processing causes a WebSphere MQ Commit call at user task sync point time.
- When WebSphere MQ persistent queues are used, an MQPUT1 call results in more processing.
- When the user task is running on the QR TCB and events are emitted synchronously to a WebSphere MQ EP adapter, more TCB switches occur.

However, all these CPU overheads are relatively low.

8.3 Monitoring and statistics

CICS monitoring collects data about the performance of all user and CICS transactions during online processing for later offline analysis. The CICS Explorer (with the CICS Performance Analyzer plug-in) can be used to view the event processing monitor data.

CICS gathers statistics data about the system resource usage and the performance of the CICS system during online processing. Reports can be generated by using the sample statistics utility program (DFH0STAT) for the event processing, event binding, capture specification and EP adapter global and resource statistics.

8.4 Problem determination

There can be times when events that are expected to be received are not captured or when you capture more events than intended. The following tools can be used to determine why the expected number of events is different from the actual number captured:

- CICS Explorer
- CICS auxiliary trace
- Statistics utility program DFH0STAT
- CICS memory dump
- INQUIRE SPI
- CICSPlex SM WUI

In this section, it is demonstrated how to use some of these tools when expected events are not captured by event processing. It is explained why unexpected events are captured and how to determine why captured data is missing, incorrect, or contains only asterisks.

Information also is provided on which steps to take to find the cause of missing events that were captured but have not arrived at their destination and if the events are to be emitted through an HTTP EP adapter to IBM Operational Decision Manager or IBM Business Monitor.

In addition to the tools listed in this section, you can use the TSQ EP adapter and the sample custom EP adapter during problem determination to check the number of events and the data that is captured. The TSQ EP adapter can be configured to write CICS event objects to a TS queue in CICS flattened event format or XML format. The sample custom EP adapter writes them to a TS queue in a simplified flattened event format.

8.4.1 Events not captured

This section gives some examples of rudimentary checks to be performed on a CICS system to determine why expected events were not captured. In addition, this section describes techniques that can be used to determine why expected events are not captured when an order process is completed in the shopping sample application.

The first check to make is that CICS can capture events. Use the CICS Explorer Event Processing view to check the EVENTPROCESS status is set to STARTED. If EVENTPROCESS is STOPPED, no events are captured. In this case, EVENTPROCESS is started.

Next, check that the event binding is installed and enabled. This can be done by using the CICS Explorer. For more information, see 5.3, “Creating and installing a bundle definition” on page 88.

The master terminal commands **CEMT INQUIRE EVENTPROCESS**, **CEMT INQUIRE EVENTBINDING**, and **CEMT INQUIRE BUNDLE** also can be used to gather this information.

If these techniques return no information for the bundle (for example, zero records returned in the Bundles view in CICS Explorer) check that the bundle was successfully deployed to the zFS. When querying bundles and event bindings, remember items stored in the zFS can have case-sensitive names.

Check that the permissions are set on the zFS directory to which the bundle is deployed to allow the files to be added or updated.

See 5.5, “Security considerations for CICS events” on page 100, to determine the security controls that might be in place for the event binding resource because these govern who has the ability to inquire on and to update the event bindings.

Check if the event binding is replaced. It is possible to replace an installed event binding by deploying a second bundle with a different name that contains an event binding with the same name as the event binding that is already installed. The consequence of this might be that expected events are not captured. For more information, see 5.4.1, “Replacing a deployed bundle” on page 99. The techniques that are described in the following section also can be used to determine if the correct event binding is currently installed on the CICS system.

Use the Bundle Definitions view in the CICS Explorer to ensure that the bundle definition contains a valid zFS directory where the bundle is stored and then initiate the install of the bundle. For more information, see 2.5, “Deploying a CICS bundle to zFS” on page 46.

If the event binding specifies a predefined EP adapter or an EP adapter set, ensure that the EP adapters or EP adapter set are installed and enabled; otherwise, events are not emitted as expected. Use the EP adapters and EP adapter Sets views in the CICS Explorer to verify that the resources are installed and in the enabled state.

For synchronous application events, check that the emission and transaction modes are applicable for the EP adapter type that was specified in the event binding because not all EP adapters can support synchronous emission with all combinations of transaction mode. Synchronous emission is not supported for system events.

There are some circumstances under which transaction abends are handled by CICS, so if the events that are missing are for unhandled transaction abend system events, check if this is the case. For example, abends issued during CICS initialization and shutdown Program List Table (PLT) programs are handled by CICS. Abends issued by a CICS web support transaction are caught and handled by the CICS web support alias program DHWBA. In addition, abends issued by CECI commands are handled by an EXEC CICS HANDLE ABEND issued by the CECI transaction.

The event binding that is used in this section (ShoppingEventBinding) contains five event specifications and each has one capture specification, as described in 4.4, “The events emitted from the sample application” on page 76.

To generate events, run the sample application to order an item, fulfill the order, and ship the order. One event is expected to be captured for each of these tasks and one SIGNAL EVENT API call to be captured, for a total of four events.

The event binding is created to emit events to the TSQ EP adapter. In this scenario, when the sample application is run, it is found that no events are captured.

After making the initial checks that were described earlier in this section and after the Event Bindings and Bundles views in CICS Explorer are refreshed, it can be seen that the event binding is not installed and that the bundle is installed but has a status of disabled. This is an indication that something might be wrong with the event binding. There is a group of messages written to the CICS message log each time a BUNDLE resource is installed.

The first message to search for is DFHRL0107, which indicates that the CICS resource lifecycle manager started to create the BUNDLE resource. If the BUNDLE resource failed to be created, or if an event binding failed to be installed successfully, there are more messages that indicate the cause of the failure.

For example, in this instance message, DFHPI1007 is seen in the CICS message log that indicates a problem with the XML data in the event binding, as shown in Example 8-1.

Example 8-1 Message DFHRL0107 followed by DFHPI1007, problem with event binding

```
DFHRL0107 I 11/27/2012 12:13:26 EPRED5 CICSUSER The CICS resource
  life-cycle manager has started to create the BUNDLE resource
  SHOPPING.
DFHPI1007 11/27/2012 12:13:26 EPRED5 00075 XML to data
  transformation failed because of incorrect input
 (XML_FORMAT_ERROR Content data found outside root element) for
  EVENTBINDING ShoppingEventBinding.
```

The next message to look for in the CICS message log is a message with the DFHEC prefix. In this example, DFHEC1003 appears, as shown in Example 8-2. This message is issued for several different reasons, as indicated in the reason at the end of the message. In this instance, the reason indicates BAD XML DATA. Message DFHRL0102 confirms that the event binding resource was not created.

Example 8-2 Messages showing an event binding was not created in CICS

```
DFHEC1003 11/27/2012 12:13:26 EPRED5 The CICS event capture component
failed to create the EVENTBINDING resource  ShoppingEventBinding in
BUNDLE SHOPPING because XML data in the event binding could not be
parsed.
DFHRL0102 E 11/27/2012 12:13:26 EPRED5 CEDA The CICS resource lifecycle
manager failed to create the resource ShoppingEventBinding and returned
with reason CALL_BACK_ERROR.
```

Examine the event binding to establish the cause of the bad XML. One cause of bad XML data is if the event binding is manually transferred from the CICS Explorer workspace on the local workstation to the zFS by using file transfer protocol (FTP) without specifying binary transfer.

Browse the event binding file in the zFS and check that the file is in ASCII format. If not, FTP the file again from the workstation specifying binary transfer or deploy the bundle from the CICS Explorer by right-clicking the bundle in the Project Explorer view and selecting **Export Bundle Project to z/OS UNIX File System**. In this instance, the bundle was installed as disabled because one part (the event binding) was not installed, so it is necessary to DISABLE the bundle again to ensure that all parts are disabled and DISCARD the bundle resource before you attempt to re-install the updated file.

When an event binding is successfully installed, the DFHRL0107 message is followed by DFHEC1001, as shown in Example 8-3, which confirms the event binding was created successfully. A separate EP adapter is created in CICS when **Use an adapter defined here** is selected on the adapter page in the event binding editor in CICS Explorer. This adapter has the same name as its associated event binding, so message DFHEP1001 also indicates the EP adapter was successfully installed.

Example 8-3 Showing a bundle, event binding and adapter installed successfully in CICS

```
DFHRL0107 I 11/14/2012 13:27:31 EPRED5 CICSUSER The CICS resource
lifecycle manager has started to create the BUNDLE resource SHOPPING.
DFHRL0125 I 11/14/2012 13:27:31 EPRED5 CICSUSER BUNDLE resource
SHOPPING is being created with BUNDLEID Shopping_Bundle and version
1.0.0.
DFHEP1001 11/14/2012 13:27:31 EPRED5 EPADAPTER ShoppingEventBinding
from BUNDLE SHOPPING installed successfully.
DFHEC1001 11/14/2012 13:27:31 EPRED5 EVENTBINDING ShoppingEventBinding
from BUNDLE SHOPPING installed successfully.
DFHRL0109 I 11/14/2012 13:27:31 EPRED5 CEDA The CICS resource lifecycle
manager has created the BUNDLE resource SHOPPING and the BUNDLE is in
the enabled state.
```

In CICS Explorer, the status of the bundle, event binding, and EP adapter can be seen in the Bundles, Event Bindings, and EP adapters views.

After the event binding and EP adapter are installed and enabled, run the shopping sample application again to order an item, fulfill the order, and ship the order. As before, four events are expected to be captured. However, only three events appear on the TS queue. Therefore, this section contains a sequence of steps to take to determine why the correct number of events are not received.

The event processing global statistics reports can provide more details about the number of events being generated. In Example 8-4, “Put events” shows the total number of captured events passed to the EP adapter through the dispatcher. “Normal events” shows the total number of normal priority events generated, as shown in Example 8-4.

Example 8-4 Extract from event processing global statistics

```
Put Events. . . . . . . . . . . . . :          3
Normal events . . . . . . . . . . . :          3
```

The event binding was configured with a TSQ EP adapter, so the line from the event processing global statistics is also relevant, as shown in Example 8-5.

Example 8-5 TSQ EP adapter data in event processing global statistics

```
Events to Tsqueue EP adapter. . . . :          3
```

However, because four events were expected but only three are shown in the statistics reports, it must be determined why one event was not generated.

In the first instance, view the event binding in the CICS Explorer event binding editor to determine if there is an obvious reason why the expected number of events are not captured by a particular capture specification. Check the filter predicates for errors that might lead to events not being captured. If the cause of the missing events cannot be determined from the event binding, generate the CAPTURESPEC resource statistics report by using the DFH0STAT sample statistics application.

CAPTURESPEC resource statistics lists details about each capture specification. Example 8-6 shows that events were received for the Order, Ship, and SendOrder events, but not for the Fulfill event (a QueryStock event is not expected because this part of the shopping sample application was not used).

Example 8-6 Capture specification resource statistics

```
CAPTURESPECs
EVENTBINDING Name . . . . . :  ShoppingEventBinding
 EPADAPTER Name. . . . . :  ShoppingEventBinding
 Enable Status . . . . . :  Enabled
                                  Events   Capture
Capturespec name  Capture point   Captured Failures  Event Name
_____________________________________________________________________
Capturefulfill    POST:PUT_CONTAINER      0        0  Fulfil
                      Current Program . .: UPDSTOCK Current Program Op . .:Equals
                      Current Transaction:          Current Transaction Op:All
                      Current Userid  . .:          Current Userid Op  . .:All
CaptureOrder      POST:LINK_PROGRAM       1        0  Order
                      Current Program . .:          Current Program Op . .:All
                      Current Transaction:          Current Transaction Op:All
                      Current Userid  . .:          Current Userid Op  . .:All
CaptureQueryEvent PROGRAM_INITIATION      0        0  QueryStock
                      Current Program . .:          Current Program Op . .:All
                      Current Transaction:          Current Transaction Op:All
                      Current Userid  . .:          Current Userid Op  . .:All
CaptureShip       POST:REWRITE            1        0  Ship
                      Current Program . .:  SHIP    Current Program Op . .:Equals
                      Current Transaction:          Current Transaction Op:All
                      Current Userid  . .:          Current Userid Op  . .:All
CaptureSendOrder  POST:SIGNAL_EVENT       1        0  SendOrder
```

```
Current Program . .:          Current Program Op . .:All
Current Transaction:          Current Transaction Op:All
Current Userid  . .:          Current Userid Op  . .:All
```

The capture specification for the Fulfill event is called Capturefulfill, and is designed to capture an event on a PUT CONTAINER API call in the program called UPDSTOCK when the container name equals OUTPUT and the stock level falls to less than 50.

So, generate and format a level 1 and 2 CICS auxiliary trace for the EC component to determine why the Fulfill event was not captured.

Within the trace, search for the group of ECEC entries that might relate to the following PUT CONTAINER API command:

```
EXEC CICS PUT CONTAINER('OUTPUT')
     FROM(UPDATE-OUT)
     END-EXEC.
```

Find the entries that show the event for the PUT_CONTAINER predicate and the container name is OUTPUT, as shown in Example 8-7. “POST-CMD” indicates the event is captured after the command completes.

Example 8-7 CICS trace entry showing the PUT_CONTAINER predicate

```
AP 3530 ECEC ENTRY - FUNCTION(EVENT_CAPTURE) PARM_LIST(27049BD8)
EVP_ADDRESS(260CFE31)
   :POST-CMD(PUT_CONTAINER)
              AB000000 00000000 01000100 27049BD8
*.......,.......................Q*
                                                  *.............
*
              00000000 000DD7E4 E36DC3D6 D5E3C1C9
*....................PUT_CONTAI*
              09C3D6D5 E3C1C9D5 C5D90000 00       *NER.............CONTAINER...
*

DD 0301 DDLO  ENTRY - FUNCTION(LOCATE) DIRECTORY_TOKEN(26E1CF20) ENTRY_NAME(26E2F197)
  LOCKING_MODE(CALLER) DIRECTORY_NAME(ECCS) NAME(....OUTPUT..........)
```

Continuing to search through the trace, it can be seen that the next ECEC trace entry confirms that the primary predicate matched. The container name is OUTPUT, as shown in Example 8-8 on page 186.

Example 8-8 CICS trace entry showing the primary predicate is matched

```
AP 353C ECEC EVENT - PRIMARY_PREDICATE MATCHED
              40404040 00000000 00000000 00000000  *....OUTPUT
............*
```

The next ECEC trace entry, shown in Example 8-9, shows that the capture specification called Capturefulfill is being processed.

Example 8-9 CICS trace entry for the Capturefulfill capture specification

```
AP 353B ECEC EVENT - FILTERING CAPTURESPEC(Capturefulfill)
EVENTBINDING(ShoppingEventBinding)
    C38197A3 A4998586 A4938689 93934040  40404040 40404040 *Capturefulfill
   E2889697 97899587 C5A58595 A3C28995  84899587 40404040 *ShoppingEventBinding
```

In this trace entry, the current program name is equal to UPDSTOCK, as was selected in the event binding (see Example 8-10).

Example 8-10 CICS trace entry showing the CURRENT_PGM predicate is true

```
AP 3534 ECEC EVENT - PREDICATE_TRUE - TESTING CURRENT-PROGRAM EQUAL
              27289350 00040000 00000000 00000000  *..>DFHECFP
..l&............*
              C5D5E36D D7C7D440                     *............CURRENT_PGM
*
                                                   *UPDSTOCK
*
                                                   *UPDSTOCK
```

However, the following ECEC trace entry (see Example 8-11) gives an indication as to why the event is not captured. The test comparing the new level of stock (in the OUTPUT container) and the predicate of the filter value is found to be false. The values in this trace entry show that the current level of stock is 45, whereas the predicate specified in the event binding is less than 5.

Example 8-11 CICS trace entry showing the LESS_THAN predicate is false

```
AP 3539 ECEC EVENT - PREDICATE_FALSE - TESTING ZONED IN PARAMETER(FROM)  LESS_THAN
              00000000 001E0802 40050080 10021005  *..>DFHECFP      ........
.......*
              40404040 40404040                    *............FROM
*
                                                   *00045
*
                                                   *00005
```

The final trace entry confirms that this event was not captured, as shown in Example 8-12.

Example 8-12 CICS trace entry showing this event has not been captured

```
AP 3531 ECEC EXIT  - FUNCTION(EVENT_CAPTURE) RESPONSE(OK) EVENTS(0)
```

To further confirm the findings within the trace, generate a CICS system memory dump by using the CEMT PER SNAP command. Format the memory dump by using Interactive Problem Control Facility (IPCS) on z/OS, with the **VERBX CICS680 'EC=3'** command.

Search in the formatted memory dump for the sub-title Event Binding Summary. The Event Binding Summary section lists details about each event binding that is installed in the CICS system, along with all of the capture specifications, filter predicates, and the number of events that are captured for each event binding.

The important pieces in the summary for the event binding in this scenario can be seen in the following examples (Example 8-13, Example 8-14, and Example 8-15). The first line in the ECEVB control block contains the event binding name, its enable status, and the name of the EP adapter that is associated with the event binding, as shown in Example 8-13.

Example 8-13 The start of the ECEVB control block

```
ECEVB: ShoppingEventBinding UserTag=V001  EPAdapter=ShoppingEventBinding
Enabled=Y
```

The ECCS: Entry shows this is the capture specification called Capturefulfill, along with the primary predicate of PUT_CONTAINER, for the container named OUTPUT, as shown in Example 8-14.

Example 8-14 Capture specification information in the ECEVB control block

```
ECCS: Capturefulfill  PUT_CONTAINER(postcmd) group/func=3416 Events=0000000000000000
Failed=00000000 Specific Primary_Predicate: CONTAINER='OUTPUT          '
```

The ECFP: Entries contain the filter predicates; CURRENT_PGM equals UPDSTOCK and data item at offset 13 (x'D') in the OUTPUT container to be less than 00005, a shown in Example 8-15.

Example 8-15 Filter predicate information in the ECEVB control block

```
Filter type    Keyword      Op  Offset   Length   …KB KM MN Value(up to 32chars)
ECFP: program        CURRENT_PGM  EQ  00000000 00000008 …00 00 00    'UPDSTOCK'
ECFP: fulldata_zoned FROM         LT  0000000D 00000005 …05 00 10    '00005'
```

So, it was thought that the capture specification was designed to check for a stock level of less than 50, but the trace entries and the event binding summary in the formatted memory dump show that what was actually specified is a predicate of “stock level of less than 5” in the event binding. This is corrected by changing the application data predicate in the filtering view for the event binding in the CICS Explorer event binding editor and saving the event binding. Re-deploy the amended event binding to the zFS and re-install the bundle into CICS, to pick up the changes made to the event binding.

8.4.2 Unexpected events captured

To determine why more events are being captured than expected, start by identifying which capture specification is capturing the unexpected events by using the statistics utility program DFH0STAT. Install the group DFH$STAT and run the STAT transaction. Press PF4 Reports and scroll to the page that contains the CAPTURESPECs report, select it, and press Enter to generate the report.

The CAPTURESPECs report (as shown in Example 8-16) lists the capture specifications for the event bindings and the number of events that are captured for each capture specification. So, if there are more than an expected number of events generated, the capture specification that is generating the extra events can be quickly identified.

Example 8-16 The CAPTURESPECs statistics report

```
CAPTURESPECs
EVENTBINDING Name . . . . . :  ShoppingEventBinding
 EPADAPTER Name. . . . . :  ShoppingEventBinding
 Enable Status . . . . . :  Enabled
                                Events   Capture
Capturespec name  Capture point          Captured Failures  Event Name
Capturefulfill    POST:PUT_CONTAINER            1        0  Fulfil
                      Current Program . .: UPDSTOCK Current Program Op . .:Equals
                      Current Transaction:          Current Transaction Op:All
                      Current Userid  . .:          Current Userid Op  . .:All
CaptureOrder      POST:LINK_PROGRAM             1        0  Order
                      Current Program . .:          Current Program Op . .:All
                      Current Transaction:          Current Transaction Op:All
                      Current Userid  . .:          Current Userid Op  . .:All
CaptureQueryEvent PROGRAM_INITIATION            0        0  QueryStock
                      Current Program . .:          Current Program Op . .:All
                      Current Transaction:          Current Transaction Op:All
                      Current Userid  . .:          Current Userid Op  . .:All
CaptureShip       POST:REWRITE                  1        0  Ship
                      Current Program . .:  SHIP    Current Program Op . .:Equals
```

```
                     Current Transaction:          Current Transaction Op:All
                     Current Userid  . .:          Current Userid Op  . .:All
CaptureSendOrder  POST:SIGNAL_EVENT              1        0  SendOrder
                     Current Program . .:          Current Program Op . .:All
                     Current Transaction:          Current Transaction Op:All
                     Current Userid  . .:          Current Userid Op  . .:All
```

For this example, use the capture specification called `Capturefulfill`. To check why the event was generated for the `Capturefulfill` capture specification, generate and format a CICS auxiliary trace that contains level 1 and 2 trace entries for the EC component. By using the information from the CAPTURESPEC statistics, search for the ECEC trace entries that relate to the capture specification named `Capturefulfill`, as shown in Example 8-17.

Example 8-17 CICS trace entry showing the capture specification name

```
AP 353B ECEC EVENT - FILTERING CAPTURESPEC(Capturefulfill)
EVENTBINDING(ShoppingEventBinding)
    C38197A3 A4998586 A4938689 93934040  40404040 40404040 *Capturefulfill
   E2889697 97899587 C5A58595 A3C28995  84899587 40404040 *ShoppingEventBinding
```

Before this entry in the trace, find the ECEC EVENT entry with PREDICATE_TRUE, as shown in Example 8-18. The predicate of “Less than 50” for the new stock level is found to be true, as the actual stock level is 45, so the event is captured.

Example 8-18 CICS trace entry showing the LESS_THAN predicate is true

```
AP 3534 ECEC EVENT - PREDICATE_TRUE - TESTING ZONED IN PARAMETER(FROM) LESS_THAN
             00000000 001E0802 40050080 10021005  *..>DFHECFP      ........ .......*
             40404040 40404040                    *............FROM                *
                                                  *00045                           *
                                                  *00050
```

The two previous examples (Example 8-17 and Example 8-18) show why the event was captured for this capture specification. The use of this technique can help to determine why unexpected events are captured. In the examples that are used here, the events were expected.

If too many events are being emitted to an EP adapter, use the EVENTPROCESS global statistics to identify the number of events that are destined for each EP adapter type, as shown in Example 8-19 on page 190.

Example 8-19 Extract from Event Processing Global Statistics showing the number of events emitted

```
Events to WebSphere MQ EP adapter . :           0
Events to Transaction EP adapter. . :           0
Events to Tsqueue EP adapter. . . . :           3
Events to Custom EP adapter . . . . :           0
Events to HTTP EP adapter . . . . . :           0
```

8.4.3 Capture data is not as expected

The payload captured along with an event can include application context data, command options, and items of application data. If the data that is captured is not the data that was expected, here are some approaches that can be used to identify the cause.

If some or all of the application data is missing from the expected payload, this might be because of an optional parameter, channel, or container not being present on the API command at the time the event is captured. It also might be because of a container, data area, or COMMAREA that was provided that is too short to hold the data item.

If all of the expected data is missing, gather a CICS auxiliary trace with level 1 and 2 data for the EC component. Format the trace and search for the phrase “UNAVAILABLE_DATA” within an ECEC trace record. The trace record contains the source of the data and ECEC trace records before this identifies the capture specification for this event. With this information, check the circumstances under which data is expected to be captured. For example, there might be occasions when the data is genuinely not available. In this instance, adjust the predicate information in the event binding to allow for this.

If data was captured with the event but it is not the correct data, this might be caused by incorrect capture information in the capture specification. This can be verified by viewing the information sources for the event binding in the CICS Explorer event binding editor. Check that the correct data types, lengths, offsets, and sources were used for the emitted business information. For example, the data might include fields of different formats, so check that all the data formats were specified correctly in the capture specification. A CICS auxiliary trace with level 1 and 2 entries for the EC component can help identify these fields.

Capturing data from the wrong parameter or container or specifying the wrong offset, length, or both in the capture specification might also lead to incorrect data being seen in the event. In this instance, a CICS dump formatted for the EC component is useful. Capture and format the memory dump as described in 8.4.1, “Events not captured” on page 180.

Search the Event Binding Summary in the dump and find the event binding and capture specification that is generating the incorrect data.

The ECCD: Entry (or entries, if there is more than one item of data being captured) for this capture specification contains the data that is being captured, its length, and where it is being captured from. Use this to identify any incorrect information in the capture specification of the event binding.

Alternatively, use the **INQUIRE SPI** commands **CAPOPTPRED, CAPDATAPRED**, and **CAPINFOSRCE** to browse the option and data filter values and the information sources that are contained in the capture specifications of the event bindings that are installed in your CICS region.

If the captured data is numeric and is too large for the formatted field length, the entire field is replaced with asterisks. Check the length that is specified in the capture specification and adjust as necessary. The event data also contains asterisks if the data that is requested to be captured is missing entirely.

9

IBM Operational Decision Manager

This chapter shows how to use the CICS event specification to create an event application in IBM Operational Decision Manager to detect event patterns.

This chapter includes the following topics:

- Scenario overview
- Building an event project in Event Designer
- Next steps

9.1 Scenario overview

IBM Operational Decision Manager can receive CICS events and perform event pattern detection. IBM Operational Decision Manager includes the following components:

- Decision Server: Used for managing decisions and detecting event patterns.
- Decision Center: Used for putting decision management in the hands of those who drive the business.

For more information about Operational Decision Manager, see this website:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp

In the Stock quote scenario, when a stock quote is requested by a customer, a CICS event is emitted. By using this kind of event, you can use the Decision Server component of Operational Decision Manager to identify a pattern; for example, where a customer submits four stock quotes within two hours. This pattern can show the customer has potential interest to purchase the stock, but has not yet made up their mind to proceed. The business can then take a proactive action to encourage the purchase, such as sending a discount offer to the customer through email.

To add to the business use case, a more complex pattern can be identified by using the Decision Server. For example, if the customer made four stock quotes but did not order the stock after one day, a discount offer might be sent to the customer. For more information about this scenario, see Figure 4-4 on page 73.

Section 9.2, “Building an event project in Event Designer” describes the steps to build the event application in the Event Designer component of Decision Server using the event specification exported from CICS Explorer.

Section 9.3, “Next steps” on page 200 provides a pointer to a Redbooks publication for further steps for using Operational Decision Manager to deploy the event application and look at the events and actions.

9.2 Building an event project in Event Designer

This section explains how to use the Event Designer to build the event application. The Event Designer is a Decision Server Events component that is based on Eclipse and supports the definition of the metadata layer that is required for business event processing. Use the Event Designer to create all the building blocks for your event application, including events, business objects, actions, and event rules.

For more information about the artifacts that can be created in an event project, review the event projects topic in the Operational Decision Manager Information Center, which is available at this website:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.events.ref%2Ftopics%2Fref_dse_eventprojects.html

9.2.1 Creating an event project

Complete the following steps to create the event project in the Event Designer:

1. In the Event Designer, select **File** → **New** → **Event Project**.
2. Enter a name for the project; for example, `QUERY-STOCK`, and click **Finish**.

9.2.2 Creating event and business objects

In the new event project, you create the event objects that represent the stock query event structure. You can create business objects from the event objects that carry information from the event received to the action that is fired by the Decision Server.

You can create the event objects manually or import an XML representation of the business event and create the business objects from this. Because you already have the XML representation of the CICS event, which is the `.xsd` file that is exported from event specification as shown in 6.3.1, “Exporting the event specification” on page 121, `QueryStock.xsd`, you can use this file to import to Event Designer and create the event objects. This file also can be found in Appendix B, “Additional material” on page 243.

Because the Decision Server uses HTTP connector to receive the events, you must make sure that the EP adapter that is used for the Stock Quote event binding is the HTTP EP adapter, and that the event format is WebSphere Business Events (WBE) when you export the event specification.

Complete the following steps to define the event object and the business object:

1. Right-click the new **QUERY-STOCK** project and select **New** → **Event**.
2. Select **Create an event based on an XML Schema**.
3. Browse to the path where the `QueryStock.xsd` file is located, click **Next**, then click **Finish.** An event called `QueryStock` is created, as shown in Figure 9-1.

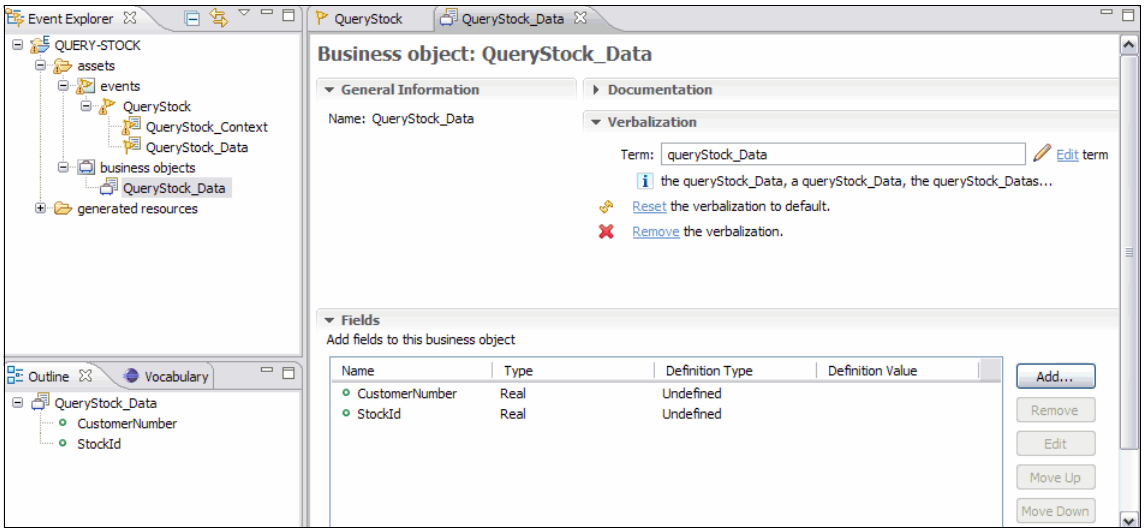


Figure 9-1 Creating QueryStock event object

4. Right-click **QueryStock_Data** under `QueryStock` event and select **Create Business Object from the Event Object**.
5. Enter `QueryStock_Data` as the business object name and click **Finish**. A business object called `QueryStock_Data` is created, as shown in Figure 9-1.

9.2.3 Creating actions and action objects

The action that starts when the event pattern is identified now can be created from the business object. Complete the following steps to create an action:

1. Right-click the `QUERY-ORDER` project and select **New** → **Action**.
2. Select **Create a blank Action** and click **Next**.
3. Enter the name `FollowUp` for the Action and click **Next**.
4. Leave the connector set to None and click **Finish**.

Complete the following steps to create the action objects:

1. Right-click the **QueryStock_Data** business object and select **Create Action Object From This Business Object**.
2. Enter a name for the action object and select the **QUERY-ORDER** project and **FollowUp** action, as shown in Figure 9-2.

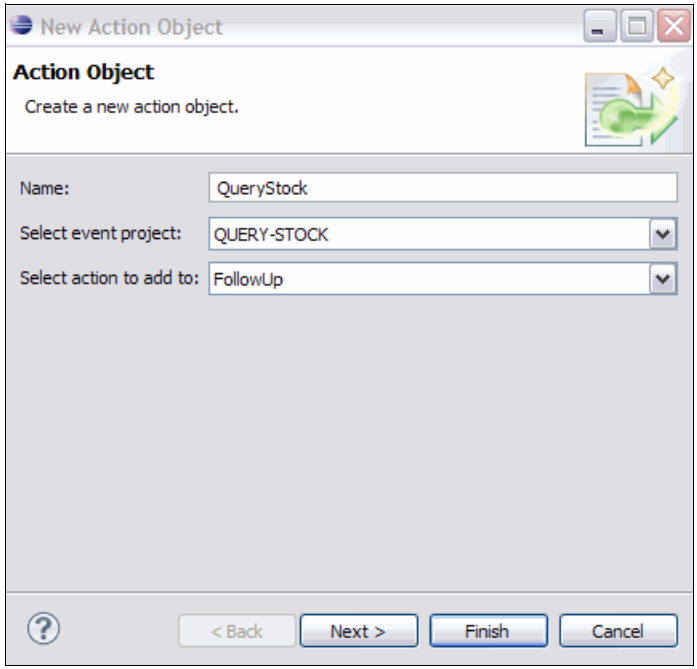


Figure 9-2 Creating the Action Object from the business object

3. Click **Finish** to create the action object.

9.2.4 Creating an event rule

You can now author the event rules to identify event patterns and perform the actions. It is a good practice to start with a simple rule and test it, then build a more complex rule on that rule.

A simple rule is shown in Figure 9-3 on page 198. On top of this simple rule, you can add a filter to filter out the customers who already ordered a stock based on the Order event, so that a discount offer does not need to be sent to these customers.

Figure 9-3 A simple event rule

For more information about building more complex rules, see the topic *Getting started with event rules* in the IBM Operational Decision Manager Information Center, which is available at this website:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.events.gs%2Ftopics%2Fwodm_dserver_events_gs.html

9.2.5 Configuring the technology connectors

The final authoring stage for the event application is to configure the event technology connectors (connectors) for the Decision Server Events run time to receive the event.

This scenario uses the HTTP event connector to receive the events from CICS.

Configuring the event connector

The Query Event is received from CICS by using the HTTP connector. Complete the following steps to configure the event connector:

1. Double-click the **STOCK-QUERY** event to open the Event editor.
2. Select the Connector tab to browse to the connector panel, as shown in Figure 9-4.

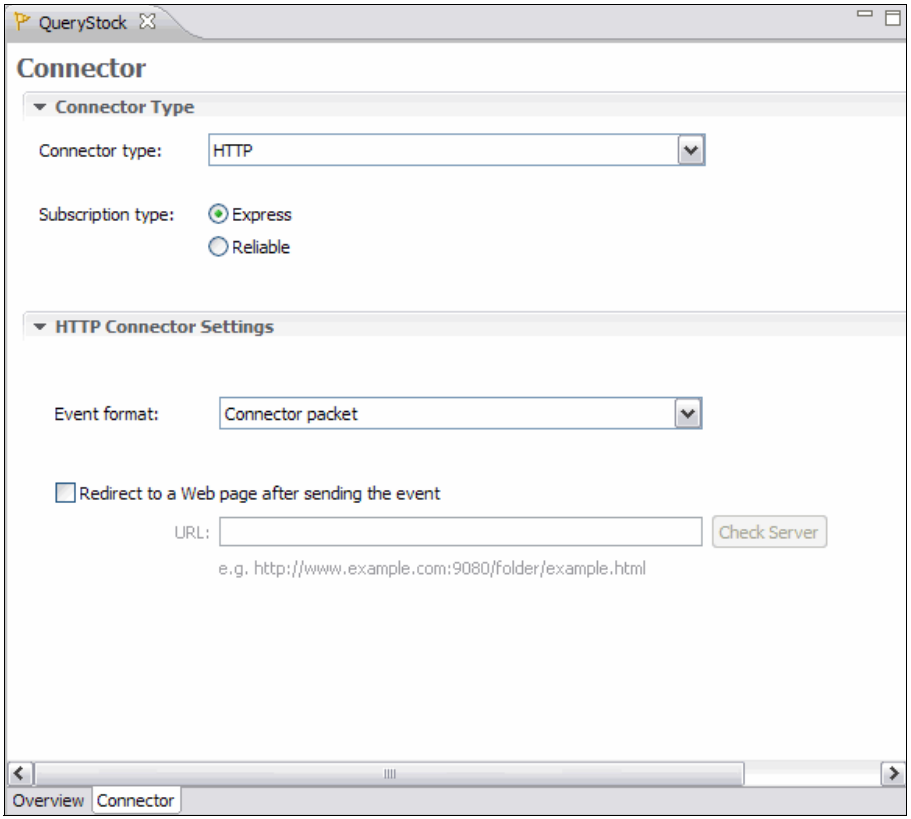


Figure 9-4 HTTP event connector

3. Select **HTTP** from the Connector Type drop-down menu as the connector type for the Request event.
4. Select **Connector packet** from the Event format drop-down menu.
5. Save the changes and close the Event editor.

At this point, the **QUERY-STOCK** project should look like Figure 9-5 on page 200, which has an event object, business object, action, action object, and an event rule.

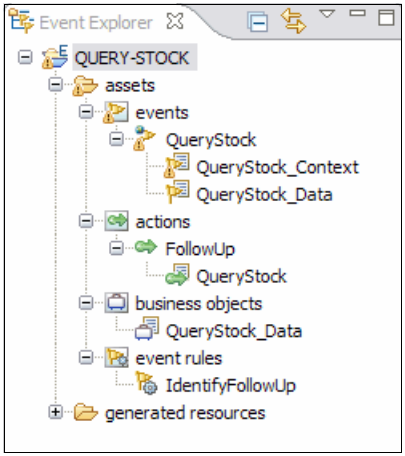


Figure 9-5 Event application project view

Configuring the action connector

To simplify testing of the scenario, you do not configure a connector for the FollowUp action. Instead, the Events tooling is used to verify that the action fired when the business rule is met.

In an actual scenario, an appropriate connector is used to send the generated action to the required system. For example, the FollowUp action can deliver an email to customer.

9.3 Next steps

You can now deploy the event application to the Decision Server Events run time on z/OS and then run the shopping sample and create some queries on the stocks. To verify that the CICS events are received by the Decision Server, you can log in to the events administrative console and look at the reports of events and actions.

For more information, see *Making Better Decisions Using IBM WebSphere Operational Decision Management*, REDP4836-00, which is available at this website:

http://www.redbooks.ibm.com/abstracts/redp4836.html

10

IBM Business Monitor

This chapter shows how to use the IBM CICS Explorer to create events and even processing (EP) adapters in the Event Binding Editor that are used by IBM Business Monitor. These events then can be exported as event schemas. After these schemas are imported into the IBM Integration Designer tool, they can be used to generate Monitor Models to be run in IBM Business Monitor.

This chapter includes the following topics:

- Scenario overview
- WebSphere MQ set up on z/OS
- Stock Level low scenario that uses WebSphere MQ EP adapter
- Scenario: Shipped order meets service level agreements by using HTTP EP adapter

10.1 Scenario overview

We use the following scenarios that are described in the Shopping sample application:

- Stock Low, which is described in section 4.3.2, “Scenario: Stock low” on page 74.
- Shipped order meets service level agreements, which are described in section 4.3.3, “Scenario: Shipped order meets service level agreements” on page 75.

The following process is used to create and emit events to IBM Business Monitor:

1. The event specification is created by using the CICS event binding editor in IBM CICS Explorer. The event specification describes the event and defines the business data that should be captured in the event. The configuration of the event binding specifies an appropriate EP adapter. In our examples, we use the following EP adapters:
   - WebSphere MQ Queue adapter with a Common Base Events format
   - HTTP adapter with a Common Base Event REST format
2. After the event binding is complete, it is saved as an XML event binding file in a CICS bundle.
3. The schemas for CICS events are exported from the adapter tab in the CICS event binding editor. The XML schema files that are created describe the event payload.
4. An IBM Business Monitor developer imports the XML schema files that were exported from the CICS event binding editor and creates a Monitor Model Application. The developer configures the Monitor Model to include the data that was captured in the event.
5. The Monitor Model is then deployed to the IBM Business Monitor server.
6. A CICS system programmer deploys the CICS bundle resource that contains the event binding to the CICS system so that the events can be detected and captured during run time.
7. When there is a match for the set of conditions that were defined in an event specification, an event is emitted and sent over the chosen EP adapter to the IBM Business Monitor server. The Monitor Model Application then processes the event.
8. A Business Space is opened and the dashboard that can be used to show the captured data is reviewed.

10.2 WebSphere MQ set up on z/OS

The steps that are described in this section are one-time setup steps that must be completed the first time the IBM Business Monitor server is configured to use CICS events through WebSphere MQ.

In our shopping application, we use the WebSphere MQ EP adapter to format and process our events.

Figure 10-1 shows how to add the WebSphere MQ loadlibs to the CICS STEPLIB and DFHRPL concatenations in our CICS startup job control language (JCL).

```
//STEPLIB  DD DISP=SHR,DSN=CICSTS.V5R1.CICS.SDFHAUTH
//         DD DISP=SHR,DSN=CICSTS.V5R1.CICS.SDFJAUTH
//         DD DISP=SHR,DSN=CEE.SCEERUN
//         DD DISP=SHR,DSN=WMQ.V7R1M0.SCSQAUTH
//DFHRPL   DD DISP=SHR,DSN=CICSTS.V5R1.CICS.SDFHLOAD
//         DD DISP=SHR,DSN=WMQ.V7R1M0.SCSQLOAD
//         DD DISP=SHR,DSN=WMQ.V7R1M0.SCSQANLE
//         DD DISP=SHR,DSN=WMQ.V7R1M0.SCSQCICS
//         DD DISP=SHR,DSN=WMQ.V7R1M0.SCSQAUTH
//         DD DISP=SHR,DSN=CEE.SCEECICS
//         DD DISP=SHR,DSN=CEE.SCEERUN2
//         DD DISP=SHR,DSN=CEE.SCEERUN
```

Figure 10-1 JCL showing WebSphere MQ loadlibs included

Put MQCONN=YES in the CICS SIT. This ensures that the MQCONN resource is installed at startup, as shown in Figure 10-2.

```
APPLID=CICSWD10
SYSIDNT=WRKS
START=INITIAL
GRPLIST=LABSIT
MQCONN=YES
```

Figure 10-2 CICS SIT showing MQCONN

We use an RDO MQCONN definition, as shown in Figure 10-3 on page 204.

```
OVERTYPE TO MODIFY                                              CICS RELEASE = 0680
 CEDA  ALter MQconn( WMQWD10  )
  MQconn          : WMQWD10
  Group           : WBM
  DEScription  ==> CONNECTION TO WMQA
  Mqname       ==> WMQA
  Resyncmember ==> Yes                 Yes ! No ! Groupresync
  Initqname    ==> CICSWD10.TRIGGER
 DEFINITION SIGNATURE
  DEFinetime      : 01/15/13 18:36:02
  CHANGETime      : 01/15/13 18:36:02
  CHANGEUsrid     : CIWD110
  CHANGEAGEnt     : CSDApi             CSDApi ! CSDBatch
  CHANGEAGRel     : 0680
```

Figure 10-3 MQCONN definition

For more information about IBM Business Monitor and how to set it up for receiving WebSphere MQ messages, see the Information Center at this website:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/topic/com.ibm.wbpm.mon.doc/home.html

Select **IBM Business Monitor, V8.0.1** → **Administering your monitoring environment** → **Integrating with event emitters**.

10.3 Stock Level low scenario that uses WebSphere MQ EP adapter

Complete the following steps to create the StockLow event in IBM CICS Explorer:

1. Switch to the Resource perspective.
2. Create a CICS bundle project by using the steps described in 6.1, “Create the CICS bundle project” on page 108 and name it `StockLowBundle`, as shown in Figure 10-4.

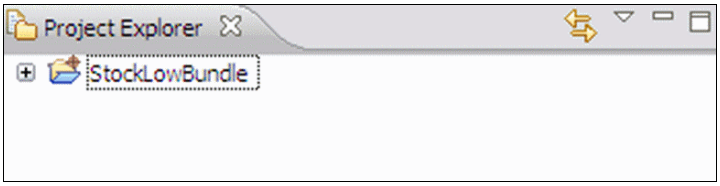


Figure 10-4 StockLowBundle

3. In the Project Explorer view, right-click the **StockLowBundle** project, then select **New** → **CICS Event Binding**.

4. Enter the file name `StockLowBinding` (as shown in Figure 10-5), then click **Finish**. The event binding editor automatically starts.

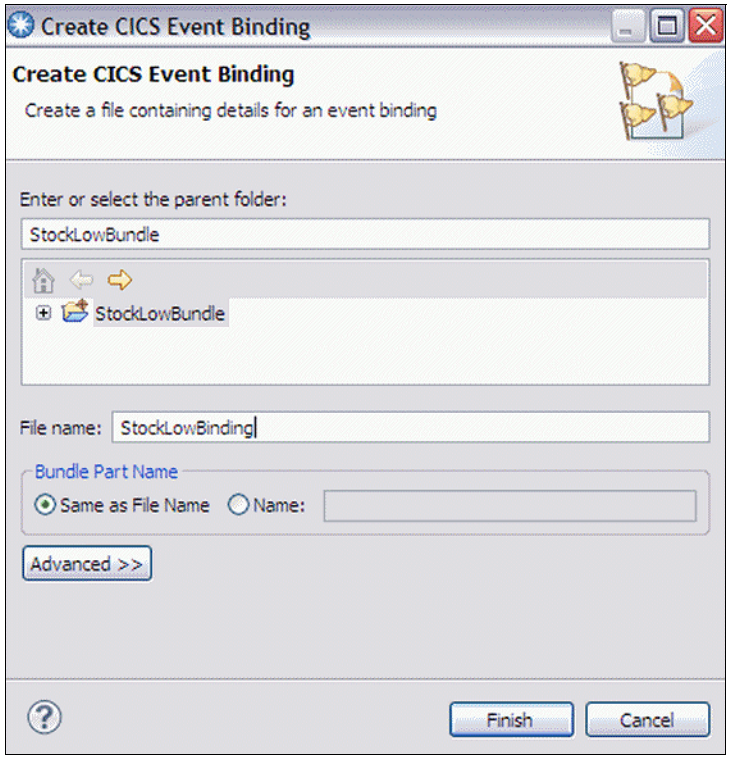


Figure 10-5 StockLowBinding

5. In the event binding editor in the Event Binding tab, enter the description `Stock below 50`, as shown in Figure 10-6 on page 206.

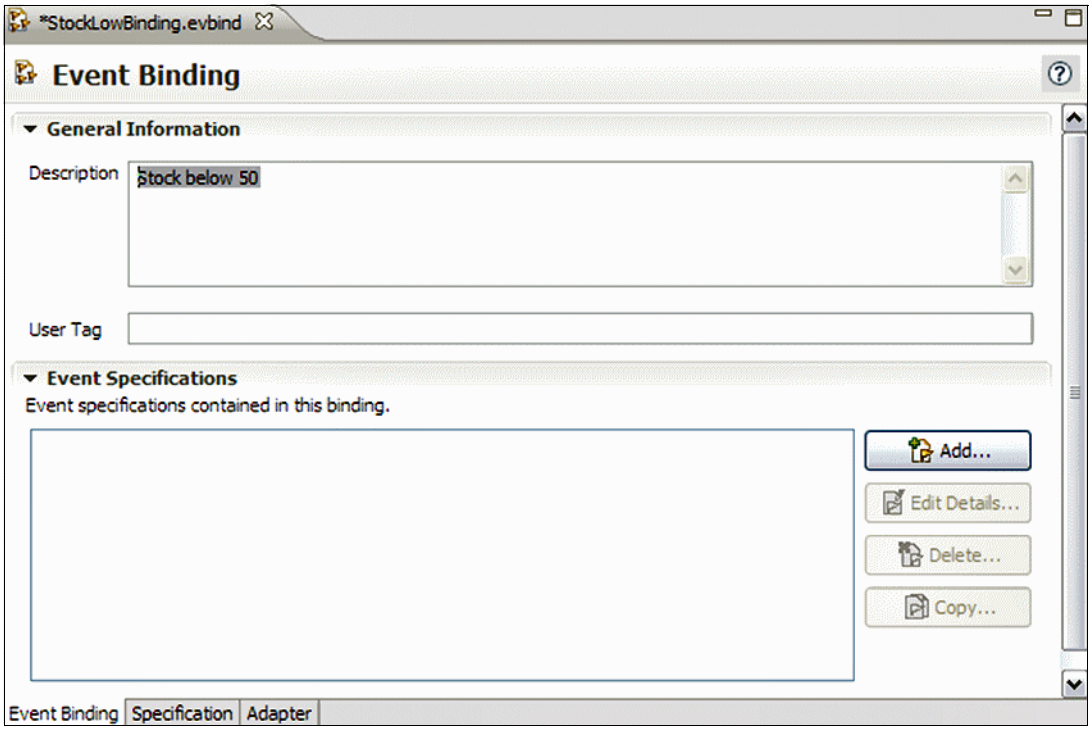


Figure 10-6 Adding the StockLowBinding specification

6. Click **Add** to add an event specification and enter the name `StockLow` and description `Stock Level Low event specification`, as shown in Figure 10-7. Click **OK**.

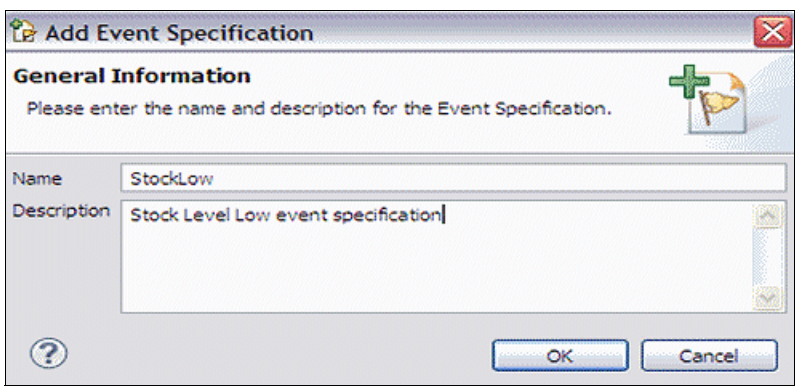


Figure 10-7 Naming the StockLow event specification

7. Click **Edit Details** to edit the event specification.

8. To emit the Stock ID, Old Stock level, and New Stock level in the event, add them in the Emitted Business Information section. Click **Add**.
9. In the Emitted Business Information window that is shown in Figure 10-8, enter the following values and click **OK**:
   - Name: StockId
   - Type: Numeric
   - Length: 6
   - Precision: 0
   - Description: The Stock ID

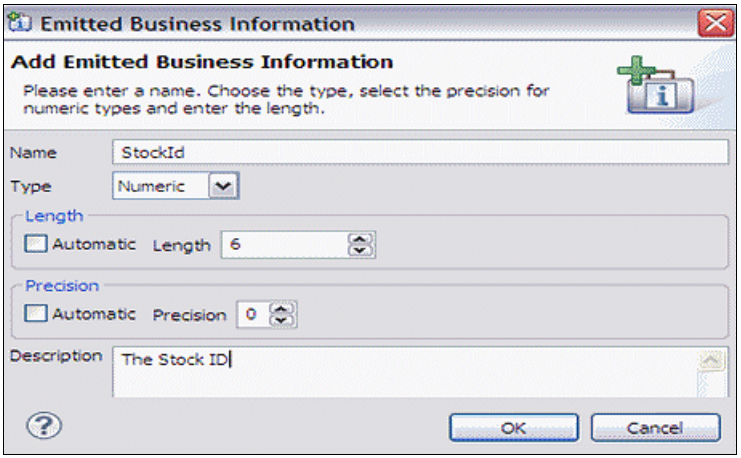


Figure 10-8 Adding the StockId business information

If you select the Automatic option, the length is set to 0, a length of 0 means capture up to the end of the data area or container. This value is useful when emitting, for example, the contents of a container regardless of its length.

10. Select **Add** again. In the Emitted Business Information window, enter the following values and click **OK**:
    - Name: OldStock
    - Type: Numeric
    - Length: 6
    - Precision: 0
    - Description: The Old Stock level
11. Select **Add** again. In the Emitted Business Information window enter the following values and click **OK**:
    - Name: NewStock
    - Type: Numeric
    - Length: 6

- Precision: 0
- Description: The New Stock level

12. In the Specifications tab that is shown in Figure 10-9, select **Add a Capture Specification**. In the next steps, we indicate when and how an event should be captured.

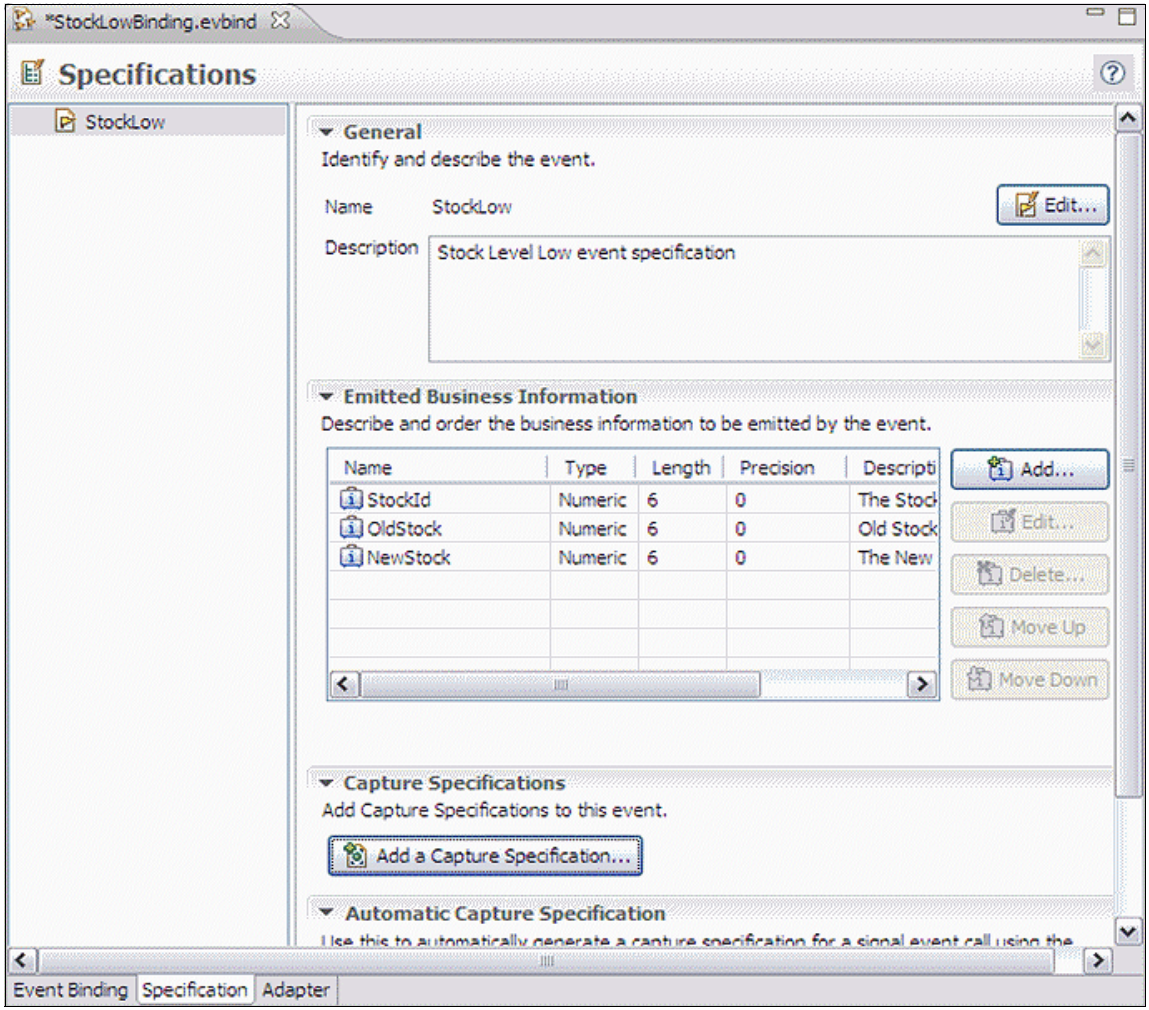


Figure 10-9 Adding the StockLow event capture specification

13. In the Add Capture Specification window, enter the following values and click **OK**:

- Name: CaptureStockLow
- Description: Capture the Stock Low event

14. Select **CaptureStockLow** in the tree on the left side. The editor shows the capture specification, including the tabs Capture Point, Filtering, and Information Sources.
15. In the Capture Point section, select **PUT CONTAINER**.
16. Select the Filtering tab. The Application Event details are shown. In the Application Context section, enter the following values:
    - Context: Current Program
    - Operator: Equals
    - Value: UPDSTOCK
17. In the Event Options section, enter the following values:
    - Name: Container
    - Operator: Equals
    - Value: OUTPUT
18. In the Application Data part, select **Add**, enter the following values, and click **OK**:
    - Operator: Less Than
    - Value: 50
    - Location: From
    - Offset: 13
    - Length: 5
19. Select the Information Sources tab (as shown in Figure 10-10) click **StockId** and then **Edit**.

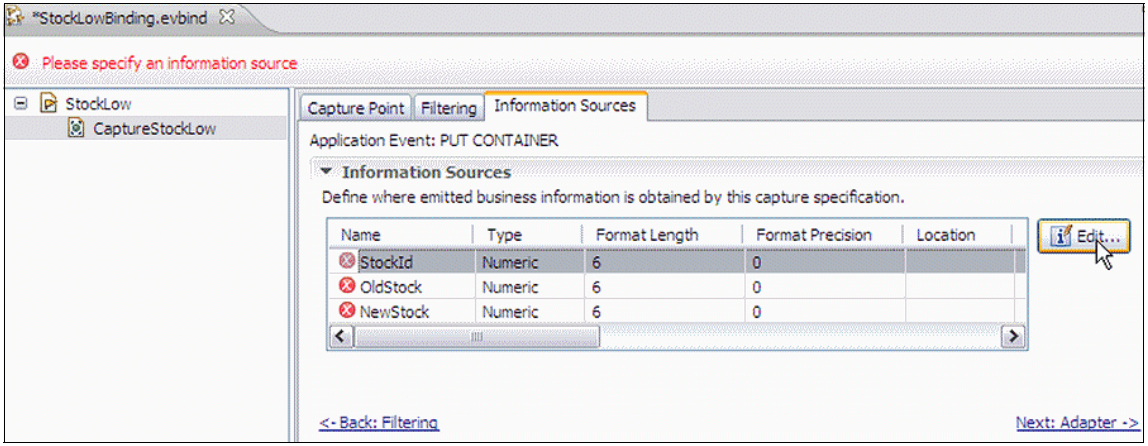


Figure 10-10 Information Sources panel

20. On the Edit Information Sources panel (as shown in Figure 10-11), select **Application Data: From** and then click **Select from imported language structure**.

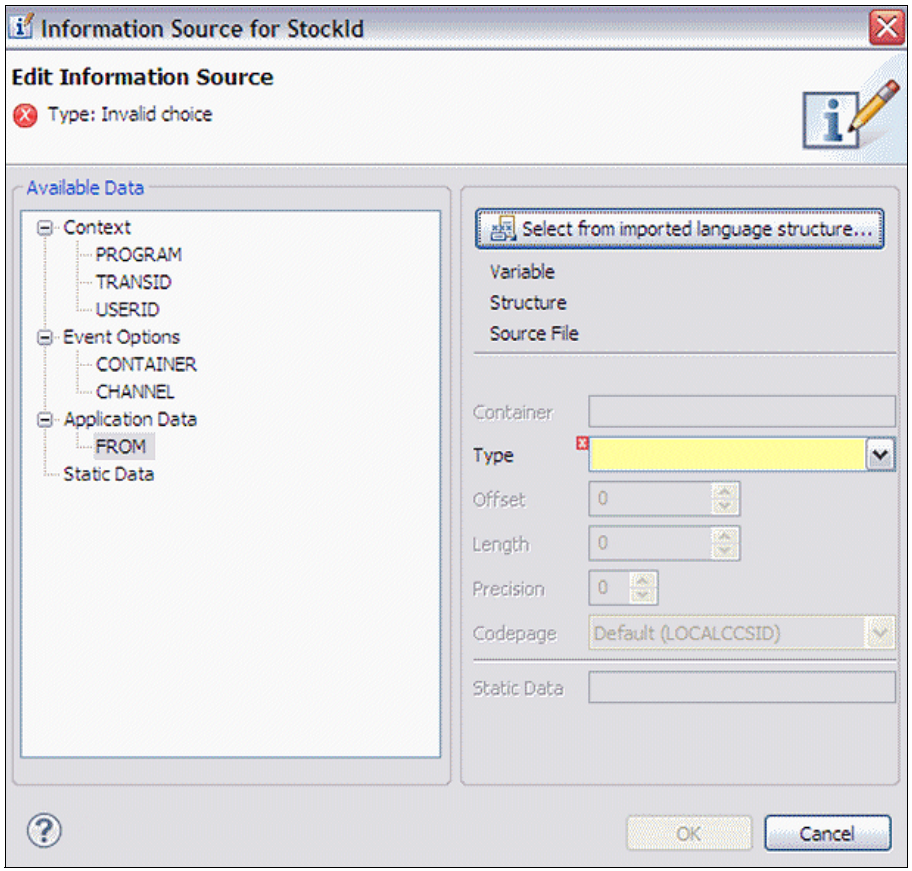


Figure 10-11 Edit Information Sources panel

21. In the Choose Source Code panel (as shown in Figure 10-12), select **COBOL** for Source Language and then click **Choose Language Structure File**.

Figure 10-12 Choose Source Code panel

22. Browse to the COBOL copy book `updstoco.cpy` and select it, click **Open**, and back on the Choose Source Code click **OK**.

23. In the Obtain data format from imported language structure panel (as shown in Figure 10-13 on page 212), click **sid** and then **OK**. Click **OK** again. Repeat the steps for OldStock and NewStock, where we set the following values:

   – OldStock: old_value
   – NewStock: new_value

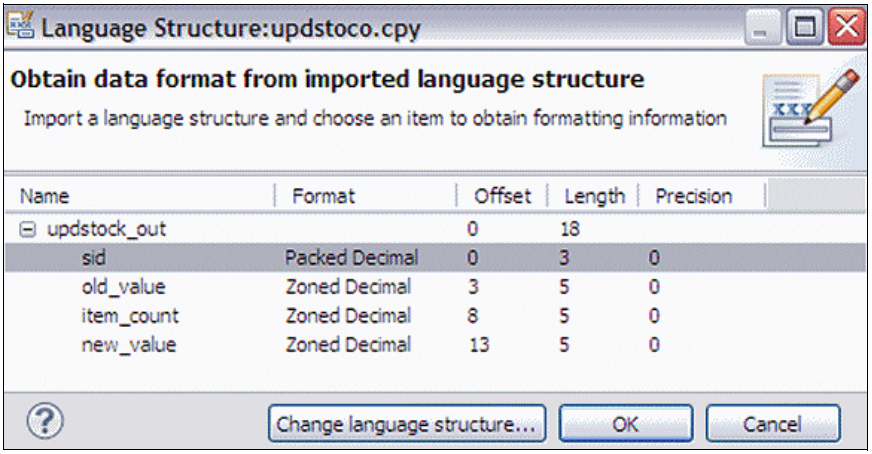


Figure 10-13 Obtain data format from imported language structure panel

24. Press Ctrl+S to save the changes. See Figure 10-14.

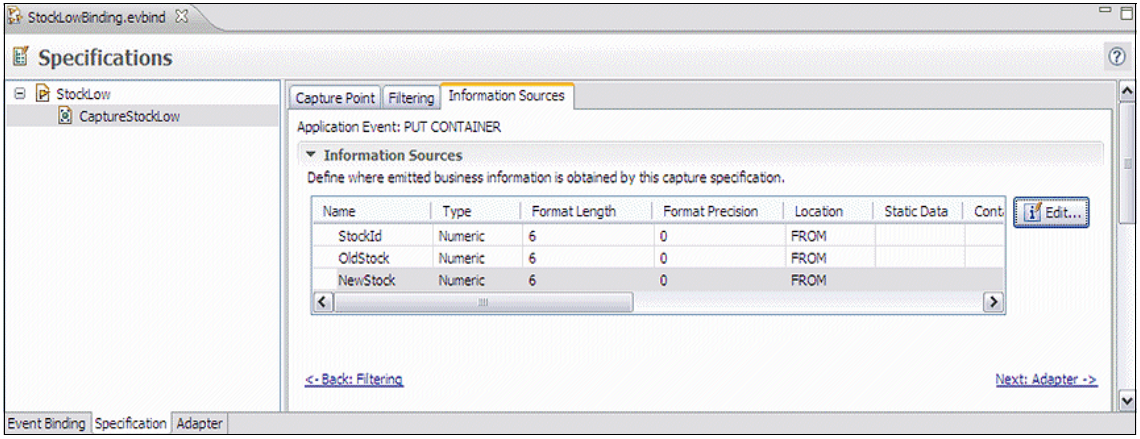


Figure 10-14 The saved StockLowBinding

10.3.1 Specifying the EP adapter

Complete the following steps to specify the EP adapter that we use:

1. In the event binding editor, select **Next: Adapter** or the adapter tab in the lower left corner of the editor.
2. Select **Use a predefined EPADAPTER resource**.

3. Enter the name `WMQ`, as shown in Figure 10-15 on page 213.

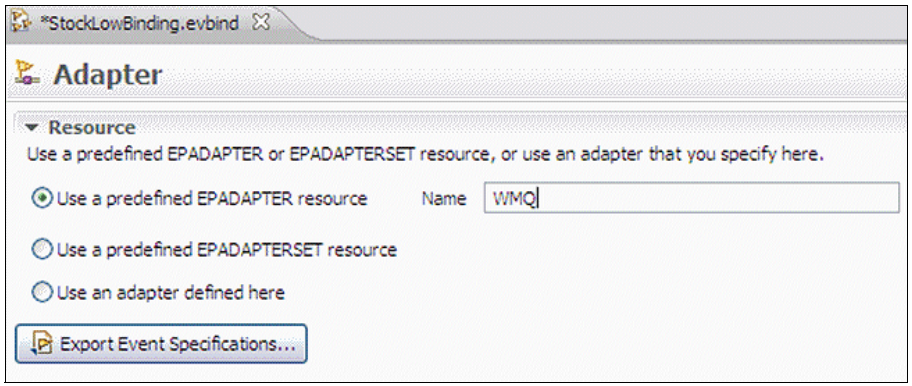


Figure 10-15 Adding a reference to the EP adapter

4. Press Ctrl+S to save the event binding.

10.3.2 Creating the WebSphere MQ adapter

Complete the following steps to create the WebSphere MQ adapter:

1. Right-click **StockLowBundle** project, then select **New** → **CICS Event Processing adapter**.
2. Enter the file name `WMQ` and click **Finish**, as shown in Figure 10-16 on page 214.

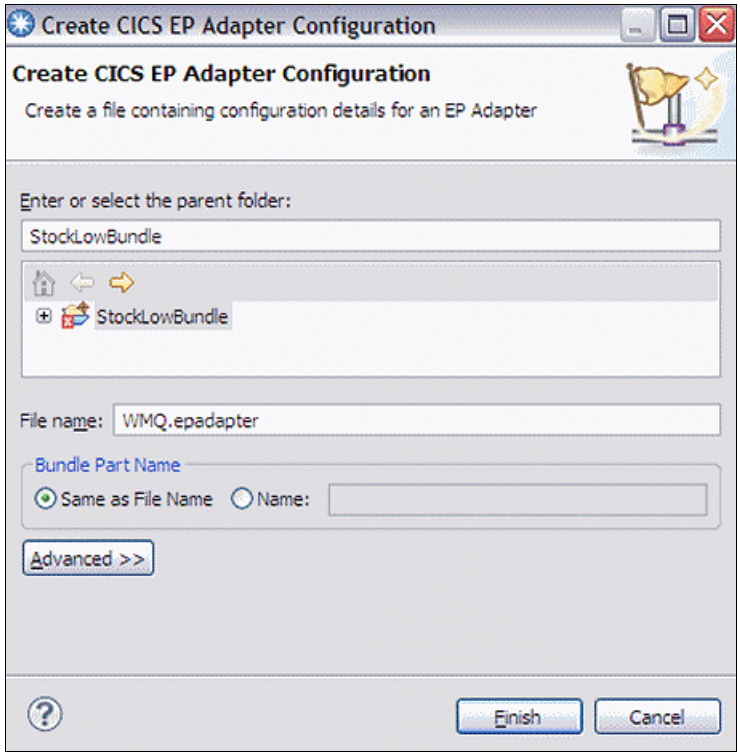


Figure 10-16 Create CICS EP adapter Configuration panel

3. In the adapter section, enter the following information as shown in Figure 10-17 on page 215, and press Ctrl+S to save:

   – Description: WebSphere MQ adapter
   – Adapter: WebSphere MQ Queue
   – Queue Name: WBM.QUEUE
   – Data Format: Common Base Event (XML)

   The queue names `WBM.QUEUE` is defined as a Remote Queue definition, which points to the destination on the system where the IBM Business Monitor is.

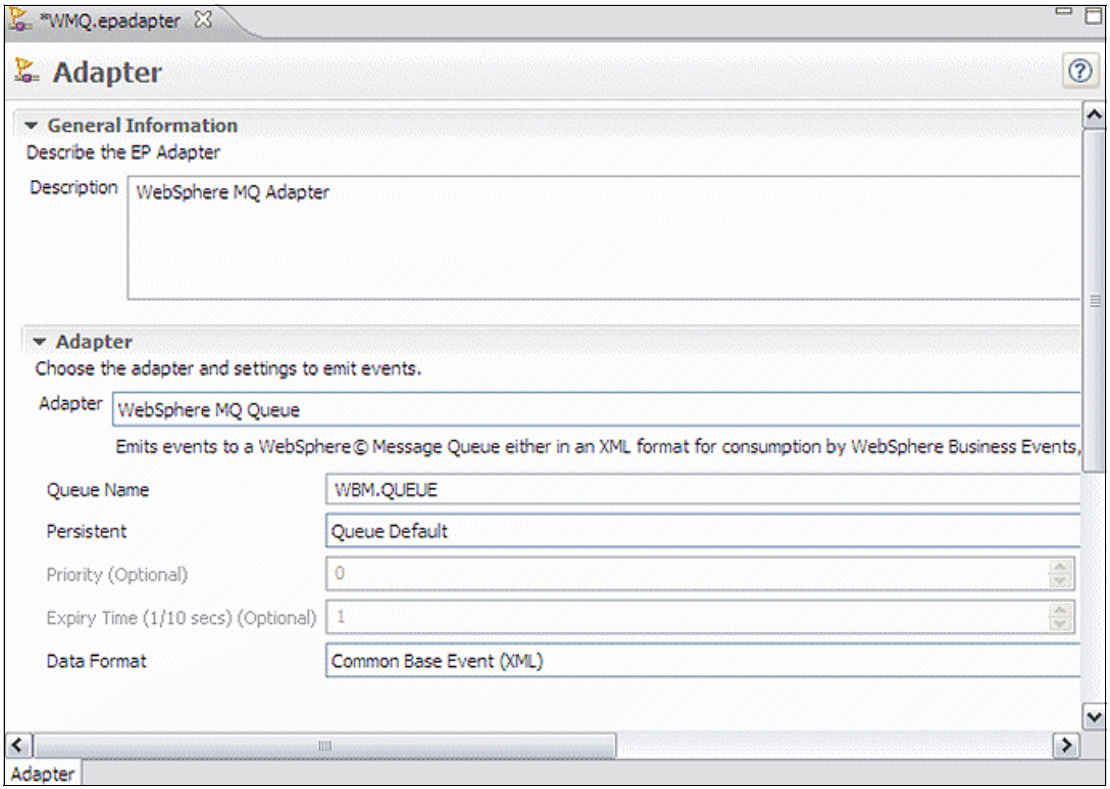


Figure 10-17 The adapter editor panel

10.3.3 Exporting the event specification

Complete the following steps to export the event specification that creates a schema that can be imported into the tooling for IBM Business Monitor:

1. Double-click the **StockLowBinding.evbind** file to open the event binding editor.

   When a WebSphere MQ EP adapter is used, the events are in the Common Base Event format. When an HTTP EP adapter is used, they are in Common Base Event REST (CBER) format. These formats include the following parts:

   – Context and environment information (known as the static schema portion)

   – Information that is specific to the event (the business information, which is known as the dynamic schema portion)

The static schema is shipped with CICS and is available on zFS along with the other supplied schemas (such as the event binding schemas) at `/usr/lpp/cists51/schemas/eventprocessing/eventformats/cics_cbe_static.xsd`. The dynamic schema is exported from the Event Binding Editor.

2. Select the adapter tab. In the resource section, select **Export Event Specifications** and select an existing directory to export the file to (see Figure 10-18), and select **OK**.

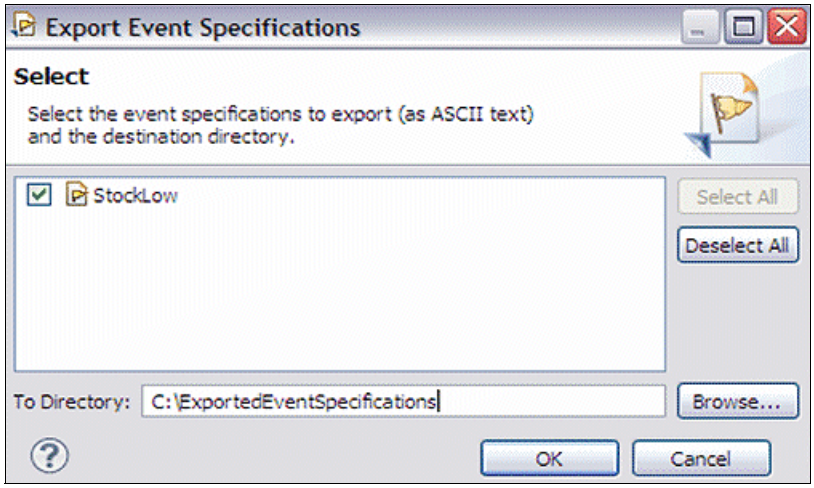


Figure 10-18 Export Event Specifications panel

If an embedded EP adapter is used (which is defined in the event binding), or the EP adapter is available in the local workspace, a schema is exported in the specified event format. If the EP adapter information is unavailable, a window offers a choice of event formats, as shown in Figure 10-19 on page 217.

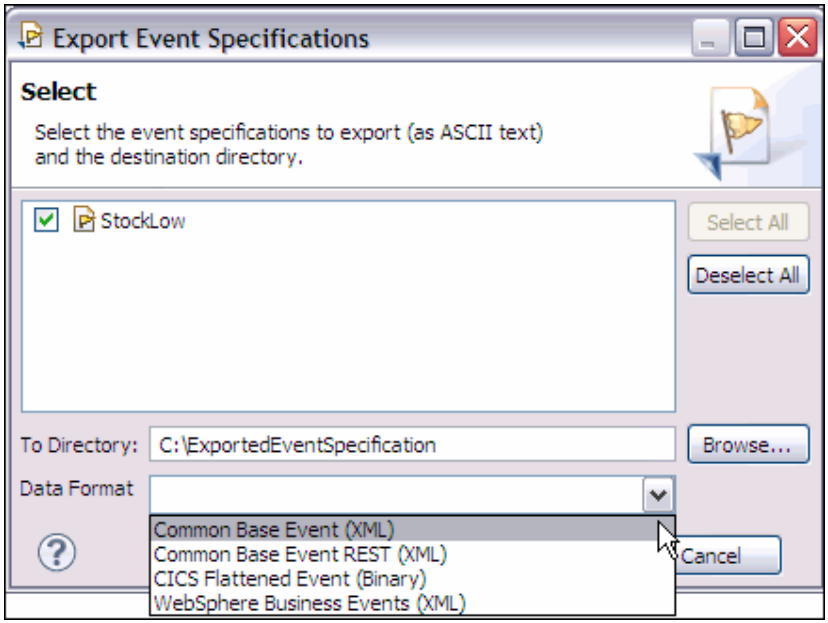


Figure 10-19 Export Event Specifications panel

3. A window displays a message that indicated Export completed, click **OK**. The file `StockLow.xsd` was created. The contents are shown in Example 10-1.

Example 10-1 StockLow.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://www.ibm.com/prod/cics/StockLow"
xmlns:tns="http://www.ibm.com/prod/cics/StockLow" elementFormDefault="qualified"
attributeFormDefault="qualified" xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="payload">
    <complexType>
      <sequence>
        <element type="string" name="StockId"/>
        <element type="string" name="OldStock"/>
        <element type="string" name="NewStock"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

4. Export the CICS Bundle project to the z/OS UNIX File System by using the process that is described in section 6.4, “Exporting the CICS bundle project” on page 123.

5. Install the CICS Bundle in CICS by using the process that is described in section 6.5, “Installing the CICS bundle” on page 125.
6. In the IBM CICS Explorer toolbar, click **Operations** → **Event Processing** → **Event Bindings** view to see the Event binding in CICS, as shown in Figure 10-20.

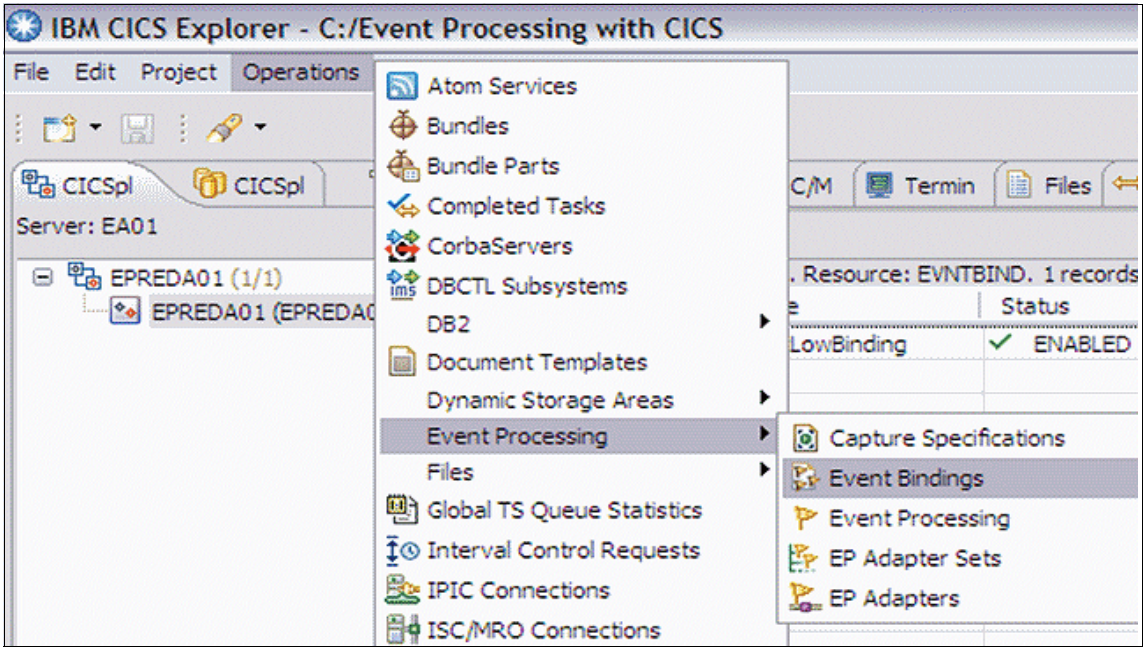


Figure 10-20 Opening the Event Bindings in the IBM CICS Explorer

The Event Binding is shown in Figure 10-21.

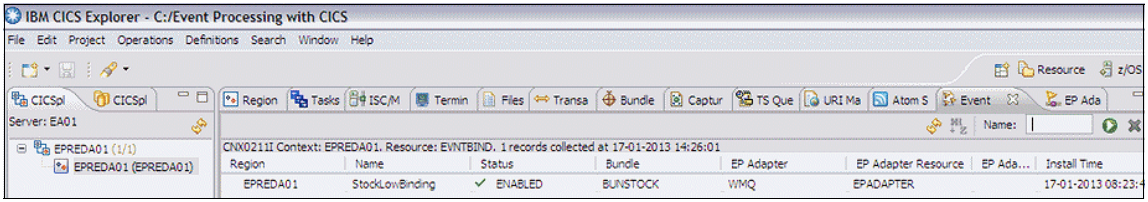


Figure 10-21 The Event Binding

7. In the IBM CICS Explorer toolbar, click **Operations** → **Event Processing** → **EP adapters** view to see the EP adapter in CICS, as shown in Figure 10-22 on page 219.

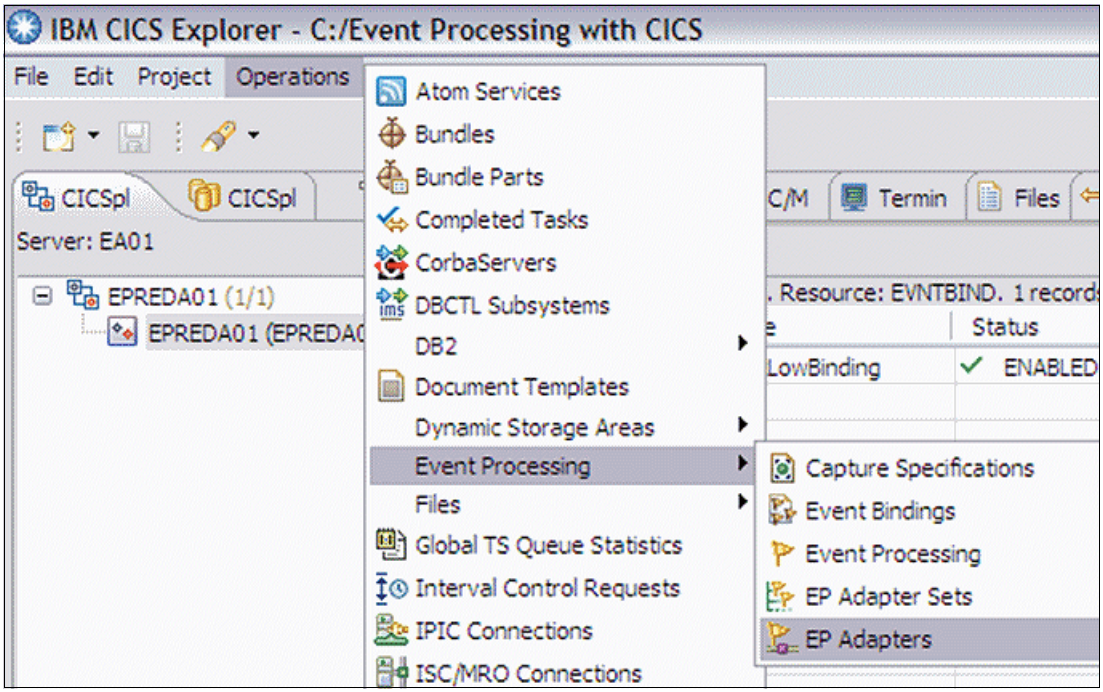


Figure 10-22 Opening the EP adapters in the IBM CICS Explorer

8. The Event Adapter is shown in Figure 10-23. Notice the data format is Common Base Event.

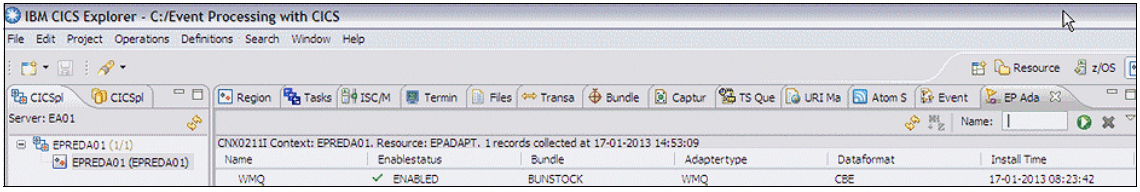


Figure 10-23 The EP adapter

10.3.4 Testing the shopping application

Complete the following steps to test the shopping application:

1. On a CICS terminal, enter the transaction ID menu and press Enter.
2. Enter the customer number `00005` and press PF2 to order an item, as shown in see Figure 10-24 on page 220.

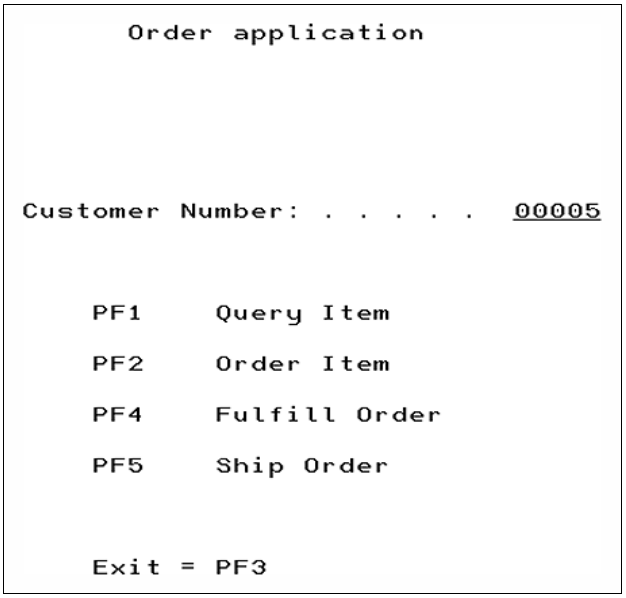


Figure 10-24 Transaction menu

3. Enter the following information, as shown in Figure 10-25, and press Enter:
   - Item number: 00010
   - Quantity: 1

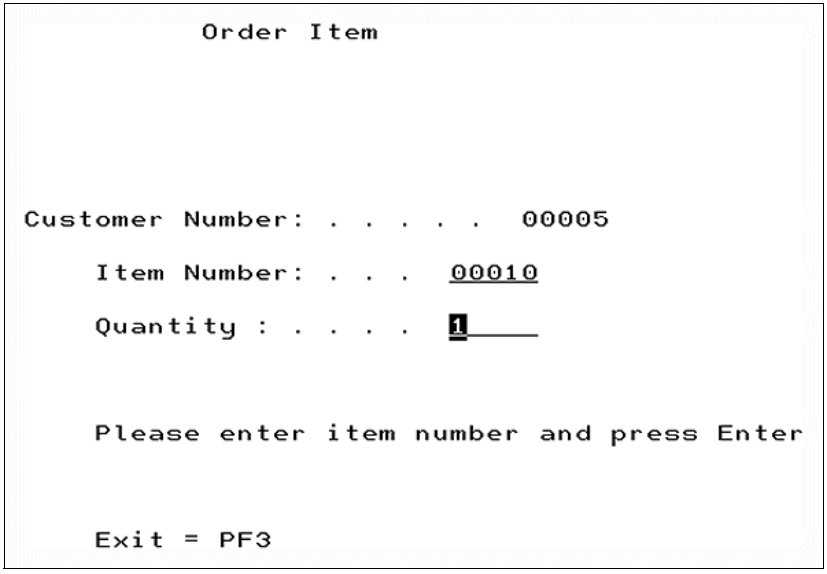


Figure 10-25 Order Item panel

The Order Confirmation panel opens, as shown in Figure 10-26.

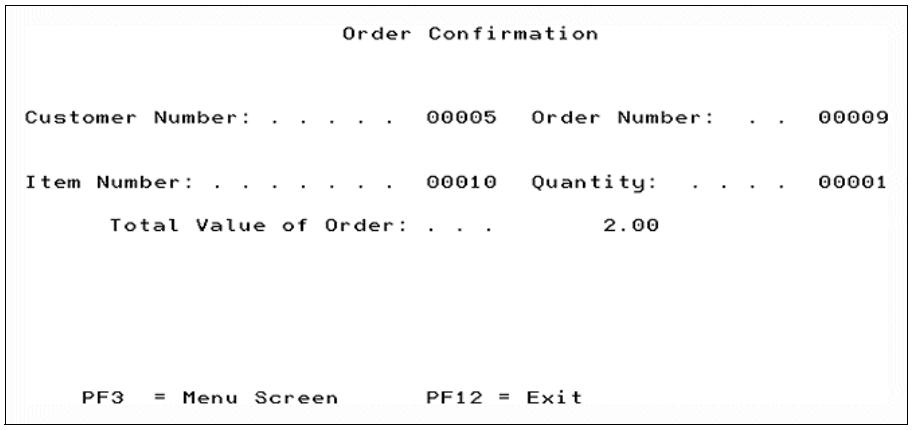


Figure 10-26 Order Confirmation panel

4. Press PF3 to return to the main menu panel.
5. Enter the Customer number 00005 and press PF4 to fulfill the order, as shown in Figure 10-27.

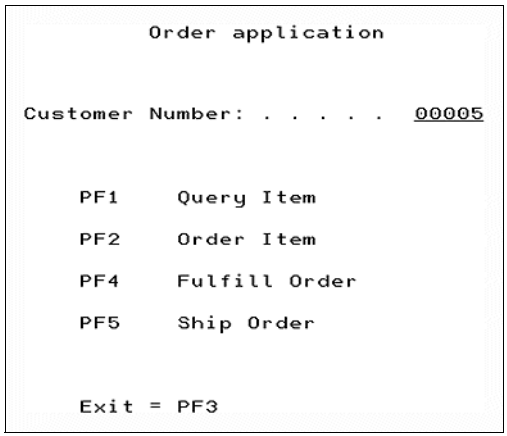


Figure 10-27 Order application panel

6. The message Number of orders fulfilled=00001 is shown, as shown in Figure 10-28 on page 222.

```
        Order application



 Customer Number: . . . . .  _____


      PF1     Query Item

      PF2     Order Item

      PF4     Fulfill Order

      PF5     Ship Order


      Exit = PF3
 Number of Orders fulfilled=00001, Total Value= 2.00, Total Items=000
0000001                                                              
```

Figure 10-28 Order Fulfilled message

7. To verify the event was emitted, use the process that is described in 6.6.1, “Verifying the event was emitted” on page 127.
8. To see the events that re-hitting the IBM Business Monitor, switch to the System where the IBM Business Monitor is installed. Enable Event Recording and verify that the event reached the Monitor, as shown in Figure 10-29.


```
<?xml version="1.0" encoding="UTF-8" ?>
- <archivedEvents:ArchivedEvents xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:archivedEvents="http://www.ibm.com/ArchivedEvents_6.2.0">
  - <archivedEvents:ArchivedEvent timeInserted="2013-01-13T15:19:57.759+00:00">
    - <CommonBaseEvent creationTime="2013-01-13T15:15:59.557+00:00" version="1.0.1">
        <sourceComponentId component="IBM CICS TS#5.1.0" componentIdType="ProductName" executionEnvironment="IBM z/OS" instanceId="MOPZT00.CICSWD10@" location="ZT01"
          locationType="Hostname" subComponent="CICS EP" componentType="http://www.ibm.com/xmlns/prod/cics/eventprocessing" />
      - <situation categoryName="OtherSituation">
        - <situationType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="OtherSituation" reasoningScope="EXTERNAL">
            <CICSApplicationEvent />
          </situationType>
        </situation>
      - <cics:event xmlns:cics="http://www.ibm.com/xmlns/prod/cics/events/CBE">
        - <cics:context-info>
            <cics:eventname>StockLow</cics:eventname>
            <cics:usertag />
            <cics:networkapplid>MOPZT00.CICSWD10@</cics:networkapplid>
            <cics:timestamp>2013-01-13T15:15:59.557+00:00</cics:timestamp>
            <cics:bindingname>StockLowBinding</cics:bindingname>
            <cics:capturespecname>CaptureStockLow</cics:capturespecname>
            <cics:UOWid>1910D4D6D7E9E3F0F04BE3C3D7F1F0F0F7F3C3D43D636820000200</cics:UOWid>
          </cics:context-info>
        - <cics:payload-data>
          - <data:payload xmlns:data="http://www.ibm.com/prod/cics/StockLow">
              <data:StockId>000010</data:StockId>
              <data:OldStock>000047</data:OldStock>
              <data:NewStock>000046</data:NewStock>
            </data:payload>
          </cics:payload-data>
        </cics:event>
      </CommonBaseEvent>
    </archivedEvents:ArchivedEvent>
  </archivedEvents:ArchivedEvents>
```


Figure 10-29 The emitted event

Now switch to the IBM Integration Designer to generate Monitor Models. For more information about how to generate Monitor Models, see this website:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/topic/com.ibm.wbpm.wid.tkit.doc/mme/developingmonitormodels.html

10.4 Scenario: Shipped order meets service level agreements by using HTTP EP adapter

In this scenario, the shipped order meets service level agreements by using the HTTP EP adapter.

10.4.1 Creating the ServiceLevelAgreement event

Complete the following steps to create the ServiceLevelAgreement event in IBM CICS Explorer:

1. Switch to the Resource perspective.
2. Create a CICS bundle project by using the process that is described in section 6.1, “Create the CICS bundle project” on page 108. Name it `ServiceLevelAgreementBundle`, as shown in Figure 10-30.

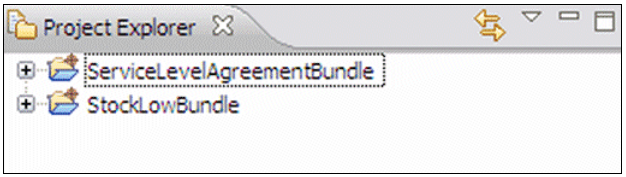


Figure 10-30 ServiceLevelAgreementBundle

3. In the Project Explorer view, right-click **StockLowBundle**, then select **New** → **CICS Event Binding**.
4. Enter the file name `ServiceLevelAgreementBinding`, as shown in Figure 10-31 on page 224. Click **Finish**. The event binding editor automatically starts.

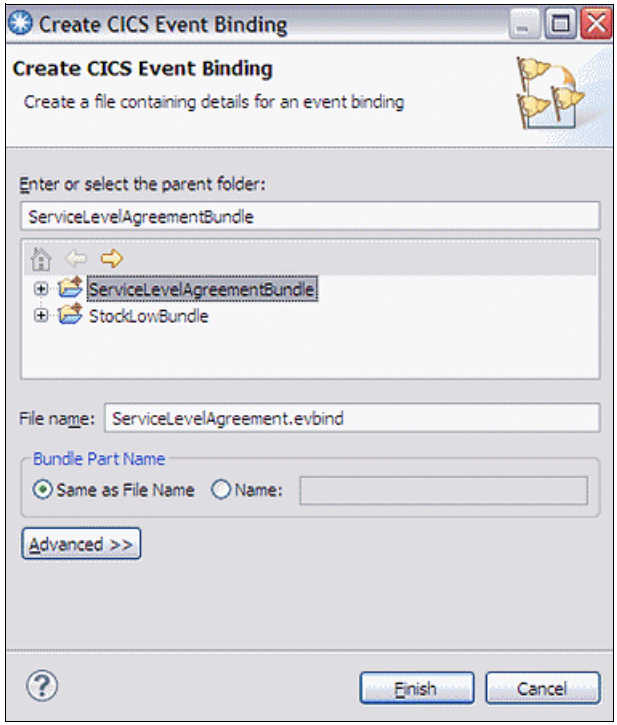


Figure 10-31 ServiceLevelAgreementBinding

5. In the event binding editor in the Event Binding tab (see Figure 10-32 on page 225), enter the description `Service Level Agreement`.

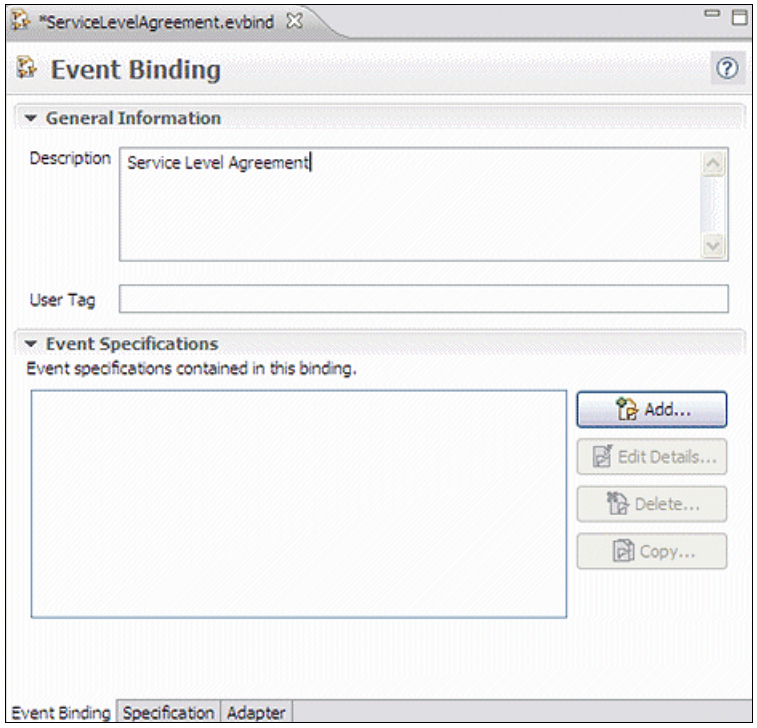


Figure 10-32 Adding the ServiceLevelAgreementBinding specification

6. Click **Add** to add an event specification. Enter the name `Order`, enter the description `Order Event Specification`, as shown in Figure 10-33. Click **OK**.

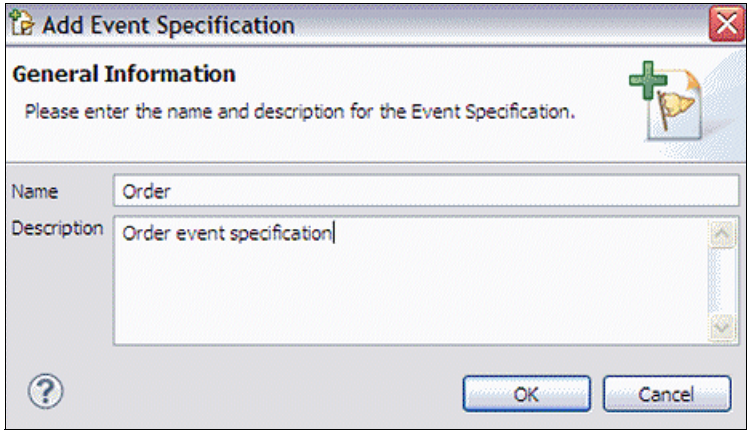


Figure 10-33 Add Event Specification

7. Click **Edit Details** to edit the event specification.
8. To emit the business information, add them in the Emitted Business Information section, as shown in Table 10-1 and Figure 10-34.

Table 10-1 Emitted Business Information

Name	Type	Length	Precision	Description
CustomerNumber	Numeric	6	0	The customer number
OrderNumber	Numeric	6	0	The order number
StockId	Numeric	6	0	The stock ID
Quantity	Numeric	6	0	The quantity

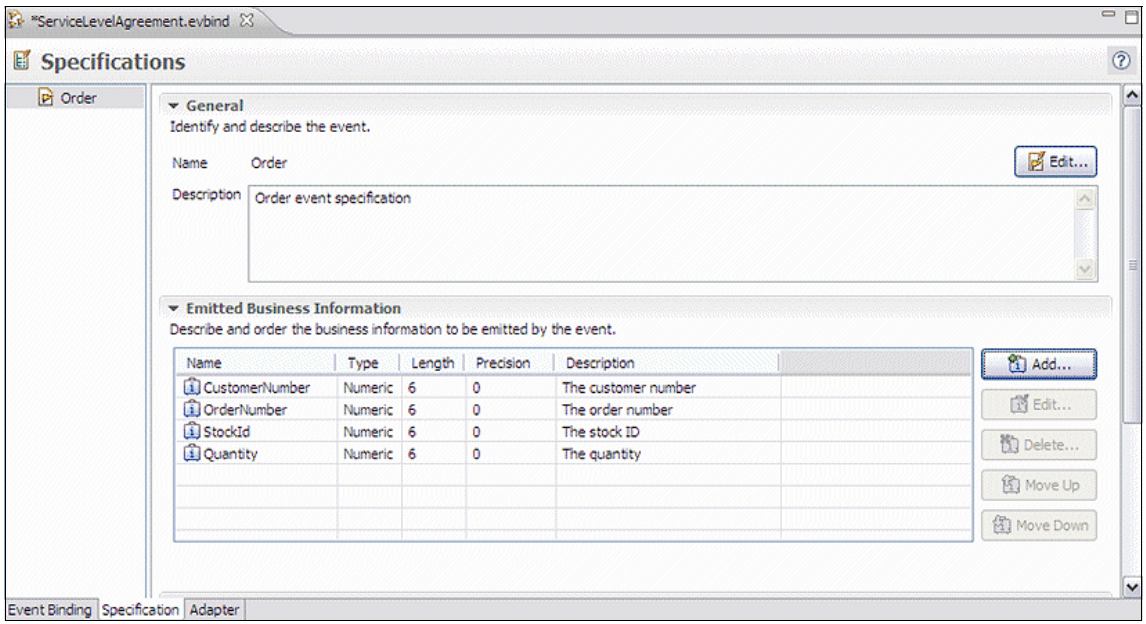


Figure 10-34 Added Business Information

9. Click **Add a Capture Specification.** In the Add Capture Specification window (see Figure 10-35 on page 227), enter the following values:
   - Name: CaptureOrder
   - Description: Capture order specification

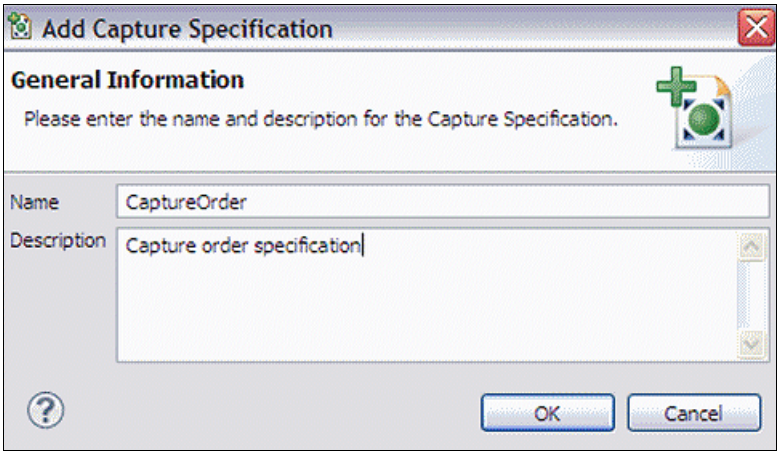


Figure 10-35 Add Capture Specification

10. Click **OK**.
11. Select **CaptureOrder** in the tree on the left side. The editor shows the capture specification, including the tabs Capture Point, Filtering, and Information Sources.
12. In the Capture Point section, select **LINK PROGRAM**. Make sure the Capture After is selected.
13. Select the Filtering tab, as shown in Figure 10-36 on page 228. In the Event Options section, set the PROGRAM to the following values:
    - Operator: Equals
    - Value: SENDORDR

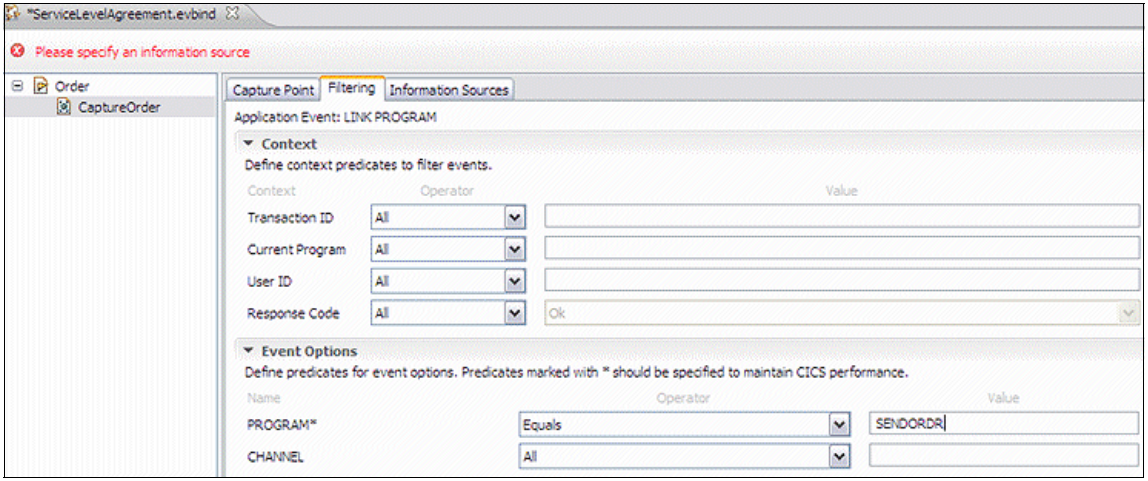


Figure 10-36 Set filtering

14. Click the Information Sources tab. In the Information Sources panel, click the information sources individually and click **Edit**. On the Edit information sources panel, click **CHANNEL** and enter the values that are shown in Table 10-2.

Table 10-2 CaptureOrder Information sources

Business Information	Container	Type	Offset	Length	Precision
CustomerNumber	EPRED-CONTAINER3	Zoned Decimal	0	5	0
OrderNumber	EPRED-CONTAINER3	Packed Decimal	15	3	0
StockId	EPRED-CONTAINER3	Zoned Decimal	5	5	0
Quantity	EPRED-CONTAINER3	Zoned Decimal	10	5	0

The result is shown in Figure 10-37 on page 229.

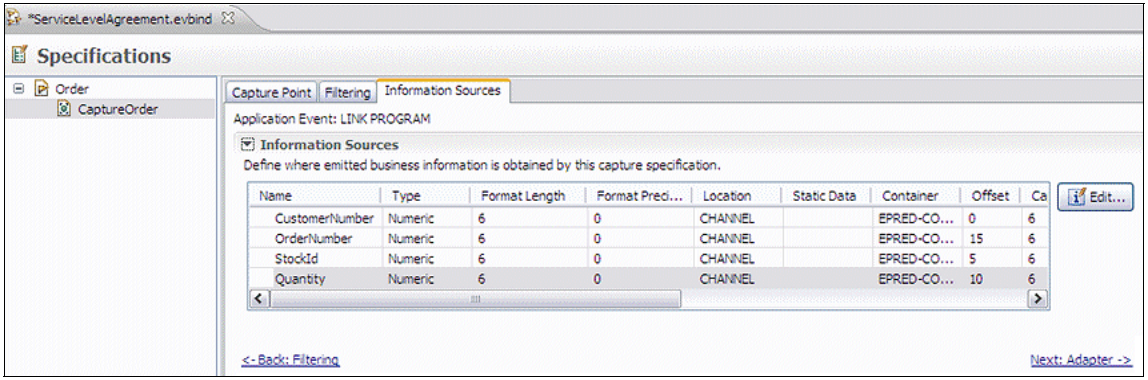


Figure 10-37 Setting information sources

15. Press Ctrl+S to save the work.

10.4.2 Defining the Ship Event specification

Complete the following steps to define the Ship Even specification:

1. Repeat step 6 on page 225 - step 14 on page 228 in the section 10.4.1, “Creating the ServiceLevelAgreement event” on page 223. Complete the following steps:
   a. On the Add Event Specification, enter the following specifications:
      - Name: Ship
      - Description: Ship event specification
   b. Click **Edit Detail**s to edit the event specification.
   c. To emit the business information, add them in the Emitted Business Information section, as shown in Table 10-3.

Table 10-3 Ship emitted business information

Name	Type	Length	Precision	Description
CustomerNumber	Numeric	6	0	The customer number
OrderNumber	Numeric	6	0	The order number
StockId	Numeric	6	0	The stock ID
Quantity	Numeric	6	0	The quantity

d. Click **Add Capture Specification** and specify the following information:

- Name: CaptureShip
- Description: Capture Ship specification

e. On the Capture Point tab, set the Capture Point to REWRITE.

f. On the Filtering tab in the Context section, set the following values:

- Context: Current Program
- Operator: Equals
- Value: SHIP
- Context: Response Code
- Operator: Equals
- Value: OK

g. In the Event Options section, set the following values:

- Name: FILE
- Operator: Equals
- Value: ORDER

h. In the Application Data section, click **Add** and add following values:

- Source: From
- Offset: 75
- Length: 1
- Operator: Equals
- Value: S

Click **OK**.

i. On the Information Sources tab, select the sources individually and click **Edit.** In the Available Data' section select **FROM** and click **Select from imported language structure**. Select **Choose Language Structure File** and find the ORDER copy book. Select **COBOL** as the Source Language, click **OK**, and select the following options:

- CustomerNumber: cid
- OrderNumber: oid
- StockId: sid
- Quantity: item-count

j. Press Ctrl+S to save the changes.

2. In this scenario, we use an HTTP adapter to send the events to IBM Business Monitor. In the event binding editor, select **Next: Adapter** or the adapter tab in the lower left corner of the editor. On the adapter panel, select **Use a predefined EPDAPTER resource** and enter `HTTP` as the name, as shown in Figure 10-38 on page 231.

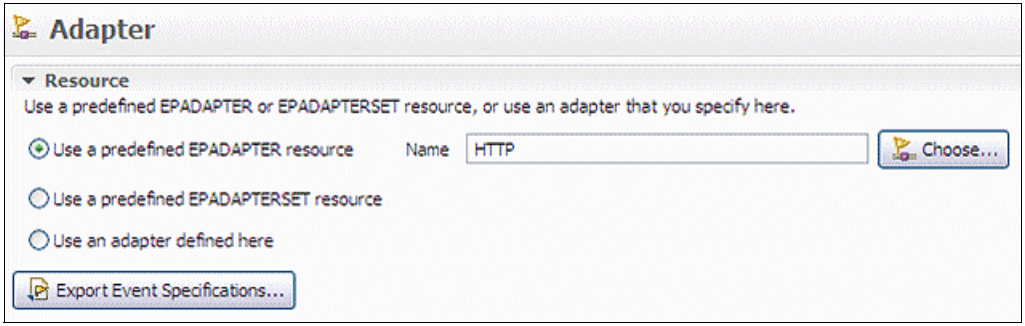


Figure 10-38 Specify EP adapter

3. Press Ctrl+S to save the event binding.

10.4.3 Creating the HTTP adapter

Complete the following steps to create the HTTP adapter:

1. Right-click the **ServiceLevelAgreementBundle** project, select **New** → **CICS Event Processing adapter**. Name the adapter `HTTP` and click **Finish**. See Figure 10-39 on page 232.

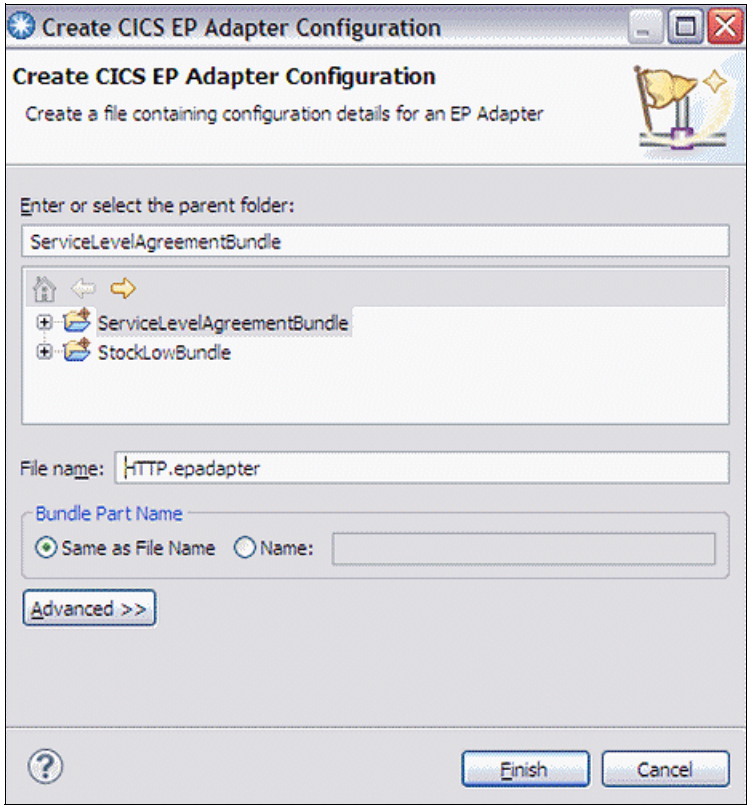


Figure 10-39 Create CICS EP adapter Configuration panel

2. In the adapter section, enter the following information as shown in Figure 10-40 on page 233, and press Ctrl+S to save:
   - Description: HTTP adapter
   - Adapter: HTTP
   - Queue Name: WBMURI
   - Data Format: Common Base Event REST (XML)

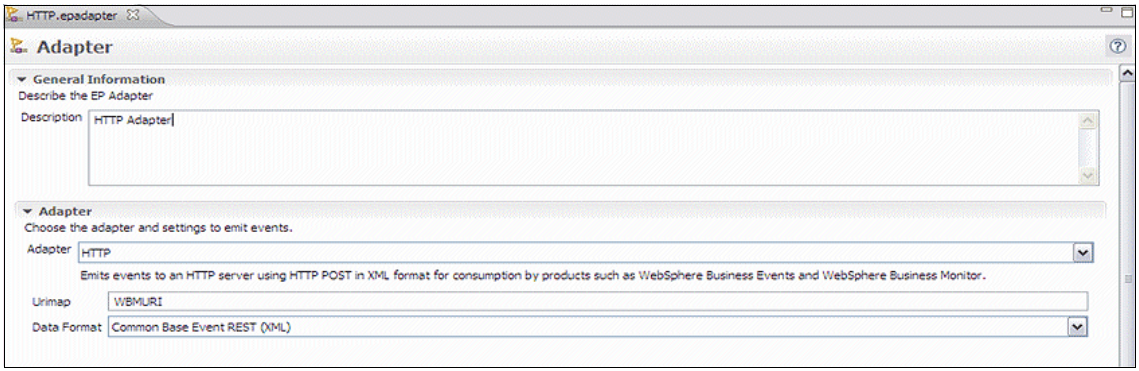


Figure 10-40 The adapter editor panel

10.4.4 Exporting the event specification

Complete the following steps to export the even specification, which creates a schema that can be imported into the tooling for IBM Business Monitor:

1. Double-click the `ServiceLevelAgreement.evbind` file to open the event binding editor.
2. Select the adapter tab. In the Resource section, select **Export Event Specifications**, as shown in see Figure 10-41.

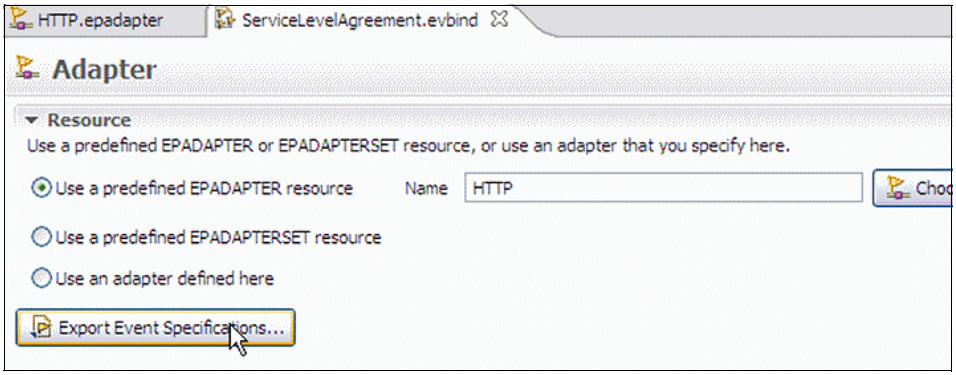


Figure 10-41 Export the event specification

3. In the next window that opens, select an existing directory to which to export the file (see Figure 10-42 on page 234). Select **OK**.

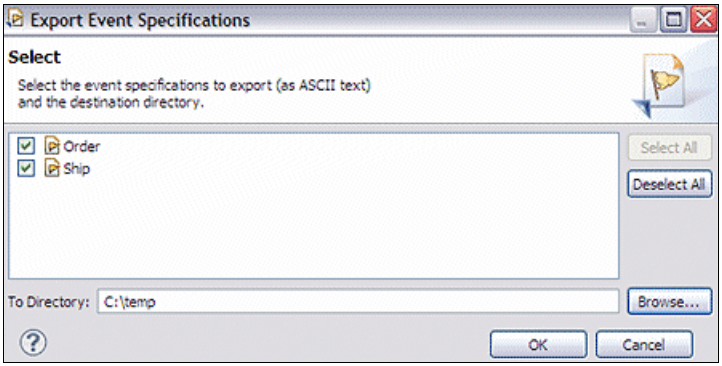


Figure 10-42 Select where to export the event specification

4. A window shows an export completed message. Click **OK**. The files `Order.xsd` and `Ship.xsd` are created. The contents is shown in Example 10-2 and Example 10-3.

Example 10-2 Order.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://www.ibm.com/prod/cics/Order"
xmlns:tns="http://www.ibm.com/prod/cics/Order"
elementFormDefault="qualified" attributeFormDefault="qualified"
xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="payload">
    <complexType>
      <sequence>
        <element type="string" name="CustomerNumber"/>
        <element type="string" name="OrderNumber"/>
        <element type="string" name="StockId"/>
        <element type="string" name="Quantity"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

Example 10-3 Ship.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://www.ibm.com/prod/cics/Ship"
xmlns:tns="http://www.ibm.com/prod/cics/Ship"
elementFormDefault="qualified" attributeFormDefault="qualified"
xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="payload">
```

```
    <complexType>
      <sequence>
        <element type="string" name="OrderNumber"/>
        <element type="string" name="CustomerNumber"/>
        <element type="string" name="StockId"/>
        <element type="string" name="Quantity"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

5. Export the CICS Bundle project to the z/OS UNIX File System by using the process that is described in section 6.4, “Exporting the CICS bundle project” on page 123.
6. Install the CICS Bundle in CICS by using the process that is described in section 6.5, “Installing the CICS bundle” on page 125.
7. To see the event binding in CICS, use the process that is described in section 10.3, “Stock Level low scenario that uses WebSphere MQ EP adapter” on page 204 step Figure 6 on page 218.

   The Event Binding is shown in Figure 10-43.

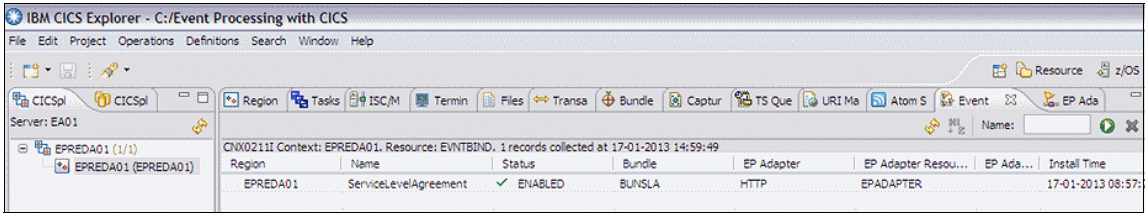


Figure 10-43 Event binding

8. Use the same method that is described in section 10.3, “Stock Level low scenario that uses WebSphere MQ EP adapter” on page 204 and Figure 6 on page 218 to see the EP adapter in CICS.

   The event adapter is shown in Figure 10-44. Notice the data format is CBER.

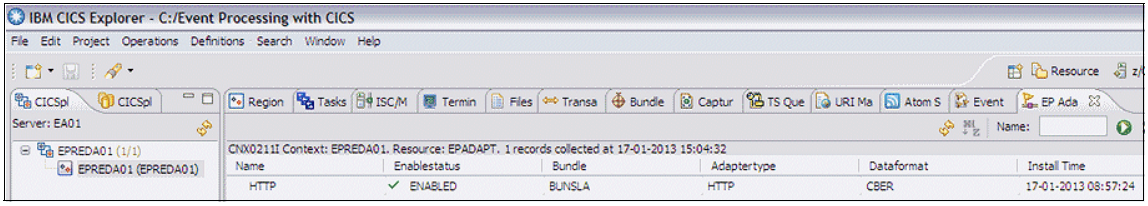


Figure 10-44 The EP adapter

9. Define the URIMAP called WBMURI that is used in the HTTP adapter, and install it (see Figure 10-45). Set other URIMAP attributes as required, such as specifying the SOCKETCLOSE parameter to use pooled HTTP connections.

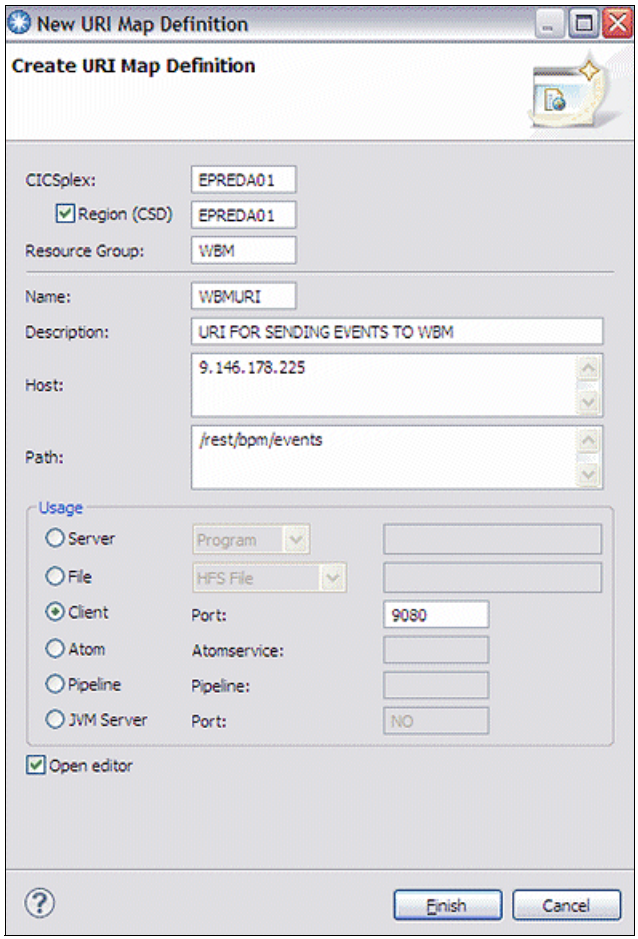


Figure 10-45 Define URIMAP for HTTP to IBM Business Monitor

10. Run the test by using the MENU transaction, as described in 10.3, “Stock Level low scenario that uses WebSphere MQ EP adapter” on page 204. Complete steps 1 - 6 in 10.3.4, “Testing the shopping application” on page 219. After step 6, press PF5 to ship the order, as shown in Figure 10-46 on page 237.

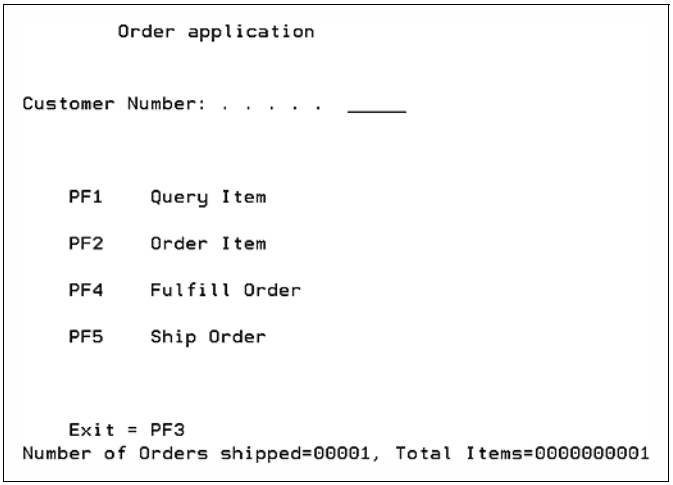


Figure 10-46 Ship Order message

To verify that the event was emitted, use the process that is described in section 6.6.1, “Verifying the event was emitted” on page 127.

11. To see the events that are hitting the IBM Business Monitor, switch to the System where the IBM Business Monitor is installed. Enable Event Recording and verify that the event reached the Monitor, as shown in Figure 10-47.

Figure 10-47 Sample of the emitted event

12. Switch to the IBM Integration Designer to generate Monitor Models.

A monitor model is an application that is used for monitoring events that are emitted by applications or processes. Monitor model applications are then run on IBM Business Monitor server to display business measures in the dashboard by using various dashboard widgets. The Monitor model editor is an editor that you use to create monitor models. It is provided as a selectable feature in IBM Integration Designer. The Monitor model can be based on BPEL processes, process applications, or any other application that can be instrumented to emit events.

For more information about how to generate Monitor Models, see this website:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/topic/com.ibm.wbpm.wid.tkit.doc/mme/developingmonitormodels.html

A

Capture points, filter predicates, and information items provided by CICS

This appendix includes tables that summarize the capture points that are provided by CICS, for each one the filter predicates available, and information items that can be captured.

This appendix includes the following topics:

- CICS application capture points
- Table A-2 lists the available system capture points as rows, and includes the following components:

CICS application capture points

Table A-1 lists the available application capture points as rows and includes the following components:

- Columns to indicate if it can be captured before or after the command is run.
- Columns for information items that can captured and emitted with the event.
- Column headings with blue shading indicate items that can filter predicates.

Table A-1 Application capture points, predicates and information sources

Application capture point	Before command	After command	Static data	PROGRAM	TRANSID	USERID	Response Code	FILE	QUEUE	QNAME	SERVICE	OPERATION	URI	CHANNEL	URIMAP	CONTAINER	PROGRAM	MAP	MAPSET	TRANSID	EVENT	FROMCHANNEL	FROM	INTO-SET	RIDFLD	CHANNEL	SCOPE	COMMAREA	FROMCHANNEL	FORMFIELD	VALUE	TEXT	Additional Filters
CONVERSE	✓	✓	✓	✓	✓	✓																	✓	✓									EIBAID, EIBCPOSN
DELETE FILE		✓	✓	✓	✓	✓		✓																	✓								
DELETEQ TD		✓	✓	✓	✓	✓			✓																								
DELETEQ TS		✓	✓	✓	✓	✓				✓																							
INVOKE SERVICE	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓											✓	✓						
LINK PROGRAM	✓	✓	✓	✓	✓	✓								✓			✓									✓		✓					
PROGRAM INIT	✓		✓	✓	✓	✓																				✓		✓					
PUT CONTAINER		✓	✓	✓	✓	✓								✓		✓							✓										
READ		✓	✓	✓	✓	✓		✓																✓	✓								UPDATE
READNEXT		✓	✓	✓	✓	✓		✓																✓	✓								UPDATE
READPREV		✓	✓	✓	✓	✓		✓																✓	✓								
READQ TD		✓	✓	✓	✓	✓			✓															✓									
READQ TS		✓	✓	✓	✓	✓				✓														✓									
RECEIVE		✓	✓	✓	✓	✓																		✓									
RECEIVE MAP		✓	✓	✓	✓	✓												✓	✓					✓									
RETRIEVE		✓	✓	✓	✓	✓																		✓									
RETURN	✓		✓	✓	✓	✓								✓						✓						✓		✓*					
REWRITE		✓	✓	✓	✓	✓		✓															✓										
SEND		✓	✓	✓	✓	✓																	✓										
SEND MAP		✓	✓	✓	✓	✓												✓	✓				✓										ALARM
SEND TEXT		✓	✓	✓	✓	✓																	✓										ALARM
SIGNAL EVENT		✓	✓	✓	✓	✓															✓	✓	✓						✓				
START		✓	✓	✓	✓	✓								✓						✓			✓			✓							
WEB READ		✓	✓	✓	✓	✓																		✓						✓	✓		
WEB READNEXT		✓	✓	✓	✓	✓																								✓	✓		✓
WRITE FILE		✓	✓	✓	✓	✓		✓															✓		✓							✓	
WRITE OPERATOR		✓	✓	✓	✓	✓																											
WRITEQ TD		✓	✓	✓	✓	✓			✓														✓										
WRITEQ TS		✓	✓	✓	✓	✓				✓													✓										
XCTL[a]	✓		✓	✓	✓	✓								✓			✓									✓		✓					

a. For RETURN capture point, the COMMAREA is only valid if specified on the command, e.g., RETURN TRANSID COMMAREA(...)

CICS application capture points

Table A-2 lists the available system capture points as rows, and includes the following components:

- Columns for information items that can captured and emitted with the event.
- Column headings with blue shading indicate items that can filter predicates.

Table A-2 System capture points, predicates and information sources

System capture point	Static data	TRANSID	USERID	DB2ID	DB2GROUPID	DB2RELEASE	FROM_CONNECTST	TO_CONNECTST	FILE	DSNAME	FROM_ENABLESTATUS	TO_ENABLESTATUS	OPENSTATUS	FROM_OPENSTATUS	TO_OPENSTATUS	ENABLESTATUS	MESSAGE_ID	INSERT1 to INSERT2	FROM_TASKS	TO_TASKS	MAXTASKS	PERCENT_MAXTASKS	TRANCLASS	FROM_ACTIVE	TO_ACTIVE	MAXACTIVE	PERCENT_MAXACTIVE	TRANSACTION	ABCODE
DB2 CONNECTION STATUS	✓	✓	✓	✓	✓	✓	✓	✓																					
FILE ENABLE STATUS	✓	✓	✓						✓	✓	✓	✓	✓																
FILE OPEN STATUS	✓	✓	✓						✓	✓				✓	✓	✓													
MESSAGE	✓	✓	✓														✓	✓											
TASK THRESHOLD	✓	✓	✓																✓	✓	✓	✓							
TRANCLASS TASK THRESHOLD	✓	✓	✓																				✓	✓	✓	✓	✓		
TRANSACTION ABEND	✓	✓	✓																									✓	✓

B

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server, which is available at this website:

`ftp://www.redbooks.ibm.com/redbooks/SG247792`

Alternatively, you can go to the IBM Redbooks web site at:

`http://www.ibm.com/redbooks`

Select **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247792.

Using the web material

The web material that accompanies this book includes the file called `SG247792.zip`, which is a compressed file of code samples.

Create a subdirectory (folder) on your workstation and extract the contents of the web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

IBM Redbooks

For more information about ordering these publications, see “How to get Redbooks” on page 246. Note that some of the documents that are referenced here might be available in softcopy only:

- *z/OS: WebSphere Business Process Management V6.2 Production Topologies*, SG24-7733
- *Business Process Management Enabled by SOA*, REDP-4495
- *Flexible Decision Automation for Your zEnterprise with Business Rules and Events*, SG24-8014-01
- I*BM Business Process Manager Version 8.0 Production Topologies*, SG24-8135-00
- *Using IBM Operational Decision Manager: IMS COBOL BMP, COBOL DLIBATCH, and COBOL MPP*, REDP-4997-00
- *Leveraging CICS Events with an ESB*, SG24-7863-00
- *Smarter Banking with CICS Transaction Server*, SG24-7815-00

Online resources

The following web sites also are relevant as further information sources:

- CICS Transaction Server for z/OS, Version 5 Release 1 information center:

  `http://pic.dhe.ibm.com/infocenter/cicsts/v5r1/index.jsp`
- CICS Explorer download:

  `http://www.ibm.com/software/products/us/en/cics-explorer`
- IBM Operational Decision Manager V8.0.1information center:

  `http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp`

- IBM Business Process Management V8.0.1 information center (including IBM Business Monitor V8.0.1)

  http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/index.jsp

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications, and additional materials, and order hardcopy Redbooks publications, at this website:

http://www.ibm.com/redbooks

Help from IBM

IBM Support and downloads:

http://www.ibm.com/support

IBM Global Services:

http://www.ibm.com/services

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Event Processing with CICS

Capture and emit events from CICS applications and system activity

Use CICS events to integrate with event consumers

Review the latest features in CICS Transaction Server V5.1

This completely refreshed IBM Redbooks® publication provides a detailed introduction to the latest capabilities for business event processing with IBM® CICS® V5. Events make it possible to identify and react to situations as they occur, and an event-driven approach, where changes are detected as they happen, can enable an application or an Enterprise to respond in a much more timely fashion. CICS event processing support was first introduced in CICS TS V4.1, and this IBM Redbooks® publication now covers all the significant enhancements and extensions which have been made since then.

CICS Transaction Server for z/OS provides capabilities for capturing application events, which can give insight into the business activities carried out within CICS applications, and system events, which give insight into changes in state within the CICS system. Application events can be generated from existing applications, without requiring any application changes.

Simple tooling allows both application and system events to be defined and deployed into CICS without disruption to the system, and the resulting events can be made available to a variety of event consumers. CICS events can amongst other things be used to drive processing within CICS, to populate dashboards that are provided by IBM Business Monitor and to search for patterns in events using IBM Operational Decision Manager.

This IBM Redbooks® publication is divided into the following parts:

Part 1 introduces event processing. We explain what it is and why you need it, and discuss how CICS makes it easy to both capture and emit events.

Part 2 of the book focuses on the details of event processing with CICS. It gives a step-by-step guide to implementing CICS events, along with the environment used in the examples.

Part 3 provides some guidance on governance and troubleshooting for CICS events, and describes how to integrate CICS events with IBM Operational Decision Manager and IBM Business Monitor.

The Appendices include additional reference information.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7792-01 ISBN 073843857X