IBM Tivoli Composite Application Manager for Transactions
V7.4.0.1
for z/OS

Installation and Configuration Guide for z/OS

SC14-7412-04

Note
Before using this information and the product it supports, read the information in "Notices" on page 85.

This edition applies to V7.4 of IBM Tivoli Composite Application Manager for Transactions (product number 5724-S79) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures v

Tables vii

About this publication ix
Publications ix
Documentation library ix
Prerequisite publications x
Accessing terminology online x
Accessing publications online x
Ordering publications x
Accessibility xi
Tivoli technical training xi
Support information xi
Conventions used in this guide xii
Typeface conventions xii
Operating system-dependent variables and paths xii

Chapter 1. Introduction 1
Overview 1
Integration with IBM Tivoli Monitoring 3
About Internet Service Monitoring 8
About Response Time 10
About Transaction Tracking 13

Chapter 2. Tivoli Enterprise Monitoring Server support on z/OS 23
Uploading catalog and attribute data files 23
Adding Response Time support for Tivoli Enterprise Monitoring Server on z/OS 24
Adding Transaction Tracking support for Tivoli Enterprise Monitoring Server on z/OS 27
Starting the hub Tivoli Enterprise Monitoring Server on z/OS 30
Setting up security for a hub Tivoli Enterprise Monitoring Server running on the z/OS operating system 31

Chapter 3. Transaction Tracking for z/OS 33
Installing and configuring 33
Software prerequisites 33
Installing and configuring Transactions Base . . 34
Transactions Base administration and operation . . 36
Transactions Container operation 38
Transactions Dispatcher operation 39

Chapter 4. CICS Tracking 43
Installing and configuring CICS Tracking 43
CICS TG to CICS tracking 46
CICS to DB2 tracking 46
CICS to IMSDB tracking 47
Tracking CICS Web Services (SOAP) traffic 47
Tracking CICS local transactions 48
Tracking included CICS function or transaction-routed interactions 48
Tracking dynamically routed CICS transactions . . 49
OMEGAMON/CICS and CICS DFHAPPL transaction umbrella names 49
TTCU 50
Dynamic maintenance procedures 51
CICS programming guide and reference 53

Chapter 5. CICS TG Transaction Tracking 57
Installing CICS TG Transaction Tracking on z/OS systems 59
Configuring the CICS TG Transaction Tracking data collector 60
Enabling CICS TG Transaction Tracking in a Gateway daemon 61
Enabling CICS TG Transaction Tracking in WebSphere Application Server 62
Enabling CICS TG Transaction Tracking in stand-alone applications 64

Chapter 6. IMS and IMS Connect Tracking 67
Installing and configuring IMS Tracking 67
Coexisting with IMS exits 69
Coexisting with IMS Tools Generic TM and MSC Message Routing Exit (GEX) 70
Installing and configuring IMS Connect Tracking 70
Using IMS Tracking with WebSphere z/OS Data Collector 71
Transaction filtering 71
IMS and IMS Connect Tracking operation 73

Chapter 7. MQ Tracking for z/OS. . . . 75
Installing and configuring MQ Tracking for z/OS . 75
Software prerequisites 75
Installing and configuring MQ Tracking for z/OS 75
MQ Tracking for z/OS administration and operation 78
Transactions MQ Container operation 79
Transactions MQ Dispatcher operation 80

Appendix. Accessibility 83

Notices 85
Trademarks 87
Privacy policy considerations 87

Glossary 89

Index 95

Figures

1. How ITCAM for Transactions integrates with IBM Tivoli Monitoring 4
2. Integration of IBM Tivoli Monitoring and other products 6
3. Internet Service Monitoring architecture . . . 9
4. How Transaction Tracking fits in to IBM Tivoli Monitoring. 16
5. IBM Tivoli Composite Application Manager for Transactions component interaction diagram . 18
6. ITCAM for Transactions on z/OS Base . . . 37
7. MQ Tracking for z/OS architecture diagram 78

Tables

1. Tivoli Monitoring and ITCAM for Transactions integration 4
2. How components integrate with IBM Tivoli Monitoring 6
3. List of protocols monitored by Internet Service Monitoring 8
4. Transaction Tracking Data Collector plug-ins 19
5. Transactions Container operator commands 39
6. Transactions Dispatcher operator commands 40
7. TTCU commands 50
8. TKANSAM library samples 53
9. Formatting for the COMMAREA 53
10. TKANMAC macros and copybooks 53
11. CICS TG Transaction Tracking - domain and transactions supported 57
12. CICS TG Transaction Tracking properties 60
13. SCYMSAMP jobs to run with specific exits 69
14. IMS and IMS Connect Tracking operation commands 73
15. Transactions MQ Dispatcher operator commands 80

About this publication

This guide provides information about installing and configuring IBM Tivoli Composite Application Manager for Transactions on z/OS.

Intended audience

This guide is for z/OS system administrators who install and configure the components of IBM Tivoli Composite Application Manager for Transactions on z/OS.

Use the information in the guides listed in "Documentation library" to further understand how to use IBM Tivoli Composite Application Manager for Transactions.

Publications

This section lists publications relevant to the use of the IBM Tivoli Composite Application Manager for Transactions. It also describes how to access Tivoli® publications online and how to order Tivoli publications.

Documentation library

The following documents are available in the IBM Tivoli Composite Application Manager for Transactions library:

- *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide*

  This guide provides information about configuring elements of IBM Tivoli Composite Application Manager for Transactions.
- *IBM Tivoli Composite Application Manager for Transactions Installation and Configuration Guide*

  This guide provides information about installing and configuring elements of IBM Tivoli Composite Application Manager for Transactions.
- *IBM Tivoli Composite Application Manager for Transactions Quick Start Guide*

  This guide provides a brief overview of IBM Tivoli Composite Application Manager for Transactions.
- *IBM Tivoli Composite Application Manager for Transactions Troubleshooting Guide*

  This guide provides information about using all elements of IBM Tivoli Composite Application Manager for Transactions.
- *IBM Tivoli Composite Application Manager for Transactions SDK Guide*

  This guide provides information about the Transaction Tracking API.
- *IBM Tivoli Composite Application Manager for Transactions User's Guide*

  This guide provides information about the GUI for all elements of IBM Tivoli Composite Application Manager for Transactions.
- *IBM Tivoli Composite Application Manager for Transactions Installation and Configuration Guide for z/OS*

  This guide provides information about using IBM Tivoli Composite Application Manager for Transactions on z/OS.

Prerequisite publications

To use the information in this guide effectively, you must know about IBM Tivoli Monitoring products that you can obtain from the following documentation:

- *IBM Tivoli Monitoring Administrator's Guide*
- *IBM Tivoli Monitoring Installation and Setup Guide*
- *IBM Tivoli Monitoring User's Guide*

If you do not have IBM Tivoli Monitoring installed already you can do a basic IBM Tivoli Monitoring installation using the IBM Tivoli Monitoring Quick Start Guide as a guide.

See IBM Tivoli Monitoring Information Center for further information.

Accessing terminology online

The IBM® Terminology website consolidates the terminology from IBM product libraries in one convenient location.

You can access the Terminology website at the following web address:

http://www.ibm.com/software/globalization/terminology

Accessing publications online

IBM posts publications for all products, as they become available and whenever they are updated, to IBM Knowledge Center.

Access IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter) using a browser.

Find supporting information on the Application Performance Management community (http://www.ibm.com/developerworks/servicemanagement/apm/index.html) and connect, learn, and share with experts.

Ordering publications

You can order many Tivoli publications online at the following website:

http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative:

1. Go to http://www.ibm.com/planetwide/.
2. In the alphabetic list, select the letter for your country and then click the name of your country. A list of numbers for your local representatives is displayed.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate most features of the graphical user interface.

For additional information, see "Accessibility," on page 83.

Tivoli technical training

For information about Tivoli technical training, see the following IBM Tivoli Education website:

http://www.ibm.com/software/tivoli/education/

Support information

If you have a problem with your IBM software, you want to resolve it quickly.

Online

Access the Tivoli Software Support site at http://www.ibm.com/software/sysmgmt/products/support/index.html?ibmprd=tivman. Access the IBM Software Support site at http://www.ibm.com/software/support/probsub.html .

IBM Support Assistant

The IBM Support Assistant is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The Support Assistant provides quick access to support-related information and serviceability tools for problem determination. The IBM Support Assistant provides the following tools to help you collect the required information:

- Use the IBM Support Assistant Lite program to deploy the IBM Support Assistant data collection tool. This tool collects diagnostic files for your product.

  Tip: When you install the IBM Support Assistant data collection tool on 64-bit systems, use a 32-bit Java Runtime Environment to ensure that data collection functions as expected.
- Use the Log Analyzer tool to combine log files from multiple products in to a single view and simplify searches for information about known problems.

For information about installing the IBM Support Assistant software, see http://www.ibm.com/software/support/isa.

Troubleshooting Guide

For more information about resolving problems, see the *IBM Tivoli Composite Application Manager for Transactions Troubleshooting Guide*.

Conventions used in this guide

This guide uses several conventions for operating system-dependent commands and paths, special terms, actions, and user interface controls.

Typeface conventions

This guide uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip**, and **Operating system considerations**).
- Keywords and parameters in text

Italic

- Words defined in text
- Emphasis of words
- New terms in text (except in a definition list)
- Variables and values you must provide

`Monospace`

- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This guide uses the UNIX system convention for specifying environment variables and for directory notation.

When using the Windows command line, replace **$***variable* with **%***variable***%** for environment variables. Replace each forward slash (**/**) with a backslash (\) in directory paths. The names of environment variables are not always the same in the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX environments.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Variables

The following variables are used in this documentation:

$CANDLE_HOME
The default IBM Tivoli Monitoring installation directory. On UNIX systems, the default directory is `/opt/IBM/ITM`.

%CANDLE_HOME%
The default IBM Tivoli Monitoring installation directory. On Windows systems, the default directory is `C:\IBM\ITM`.

$ALLUSERSPROFILE
On UNIX systems, `/usr`

%ALLUSERSPROFILE%
On Windows 7 and 2008, the default directory is `C:\ProgramData`.

Chapter 1. Introduction

IBM Tivoli Composite Application Manager for Transactions (ITCAM for Transactions) consists of several components which measure internet services and response times, and track transactions, enabling you to identify and isolate problems in your information technology environment. ITCAM for Transactions integrates with the Tivoli Enterprise Portal in IBM Tivoli Monitoring enabling you to manage your entire enterprise with a single user interface.

ITCAM for Transactions includes the following components:

- Internet Service Monitoring
- Response Time
- Transaction Tracking

Overview

ITCAM for Transactions delivers a comprehensive, unified transaction tracking management system that runs on a single, consolidated infrastructure with a tightly integrated user interface. Because problem isolation in today's complex IT environments can often take hours or days and can result in lost time, lost revenue, and low customer satisfaction, ITCAM for Transactions helps you rapidly isolate problem components which speeds up diagnosis and service restoration before poor customer experiences can directly affect revenue.

ITCAM for Transactions offers the following benefits:

- Integrates with the Tivoli Enterprise Portal in IBM Tivoli Monitoring so you can manage the entire enterprise with a single user-interface and quickly navigate views. This integration means that you do not need to learn multiple tools with different user interfaces so you can experience a faster return on investment.
- Provides several components for measuring internet services and response times, and tracking transactions, so that you can identify any problems when they occur or even *before* they occur, and isolate the problems to a specific part of your IT environment. ITCAM for Transactions also integrates with IBM Tivoli diagnostic tools such as Tivoli Business Service Manager, ITCAM for Application Diagnostics, and Tivoli OMEGAMON XE so that you can potentially diagnose and analyze any problems and then hand the details to the appropriate specialist to take corrective action.
- Provides the Application Management Console, so you can have an immediate view of your entire enterprise as a physical mapping of platforms, systems, monitoring agents, and monitored resources that shows operational status with links to the underlying component workspaces.
- Reduces the costs for IT lifecycle operations, support, and development through proactive, real-time, and automated problem resolution by providing an end-to-end view of services, transactions, and associated resources across platforms and subsystems.
- Reduces the time between problem identification and problem resolution by automatically identifying problem components in a transaction.
- Reduces the need for costly and hard-to-find subject matter experts.
- Increases revenue and customer satisfaction by maintaining service level agreements.

- Increases the performance and availability of business-critical applications, including portal and service-oriented architecture (SOA) based technologies.
- Provides role-based user interfaces so you can provide the right level of information to the right user for help with quick problem identification, seamless hand off, and problem resolution.
- Integrates performance, availability, and problem identification information with several other IBM Tivoli products to help deliver even greater value. You can use response time information with the following products:
  - IBM Tivoli Performance Analyzer to identify trends.
  - IBM Tivoli Business Service Manager to identify the impact to overall business services.
  - IBM Tivoli Provisioning Manager to take provisioning actions to help prevent SLA breaches.
  - IBM Tivoli Monitoring to determine if resource monitors (for CPU, memory, disk utilization, and so on) reveal the cause of problems. See "Integration with IBM Tivoli Monitoring" on page 3.

  See "Integration with other products" on page 6.

ITCAM for Transactions includes the following components:

- Internet Service Monitoring, which provides the tools to identify availability and response time problems and monitors to test the availability and performance of various internet services, including monitoring web sites, web-based e-commerce applications, and electronic mail (as well as the underlying services such as DNS, LDAP, and SMTP on which those services rely).

  See "About Internet Service Monitoring" on page 8 for more information.
- Response Time, which focuses on the end user experience, both real and simulated, by monitoring web transactions, playing back recorded scripts, and real user desktop experiences. Response Time includes the following components:
  - Application Management Console and Application Management Configuration Editor
  - Robotic Response Time
  - Web Response Time

  See "About Response Time" on page 10 for more information.
- Transaction Tracking, which delivers an end-to-end view of response times across systems to quickly help isolate the cause of response time and availability problems. Transaction Tracking includes the following components:
  - Transaction Collector
  - Transaction Reporter
  - Transaction Tracking API
  - CICS TG Transaction Tracking
  - CICS TXSeries Data Collector
  - Data Collector for WebSphere Message Broker
  - MQ Tracking
  - Tuxedo Tracking
  - WASTT
  - Transaction Tracking for z/OS
    - Transactions Base
    - CICS TG Transaction Tracking

- CICS Tracking
- IMS Tracking
- MQ Tracking for z/OS

– Transaction Tracking also integrates with:
- Robotic Response Time
- Web Response Time
- ITCAM for Application Diagnostics
- ITCAM for J2EE
- ITCAM for SOA
- Tivoli Business Service Manager
- Optim Performance Manager
- WebSphere Application Server
- IBM HTTP Server
- IBM Tivoli OMEGAMON XE for CICS
- IBM Tivoli OMEGAMON XE for IMS
- IBM Tivoli OMEGAMON XE for Messaging
- Monitoring Agent for Microsoft .NET Framework
- Monitoring Agent for Microsoft Internet Information Services
- Monitoring Agent for Active Directory

See "About Transaction Tracking" on page 13 for more information.

Integration with IBM Tivoli Monitoring

IBM Tivoli Monitoring is the base software for ITCAM for Transactions. IBM Tivoli Monitoring provides a way to monitor the availability and performance of enterprise systems from one or several designated workstations. It also provides useful historical data for tracking trends and troubleshooting system problems.

You can use IBM Tivoli Monitoring to do the following tasks:

- Monitor for exception conditions on the systems that you are managing by using predefined situations or custom situations
- Establish performance thresholds
- Investigate the causes leading to an exception condition
- Gather comprehensive data about system conditions
- Perform actions, schedule work, and automate manual tasks
- Using the operating system agents:
  – Provide basic performance data about operating systems and hardware to Tivoli Enterprise Management Agents
  – Provide remote functions for the Tivoli Enterprise Management Agents
  – Provide Proxy Agent Services

Figure 1 on page 4 illustrates the relationship between the IBM Tivoli Monitoring and ITCAM for Transactions components.

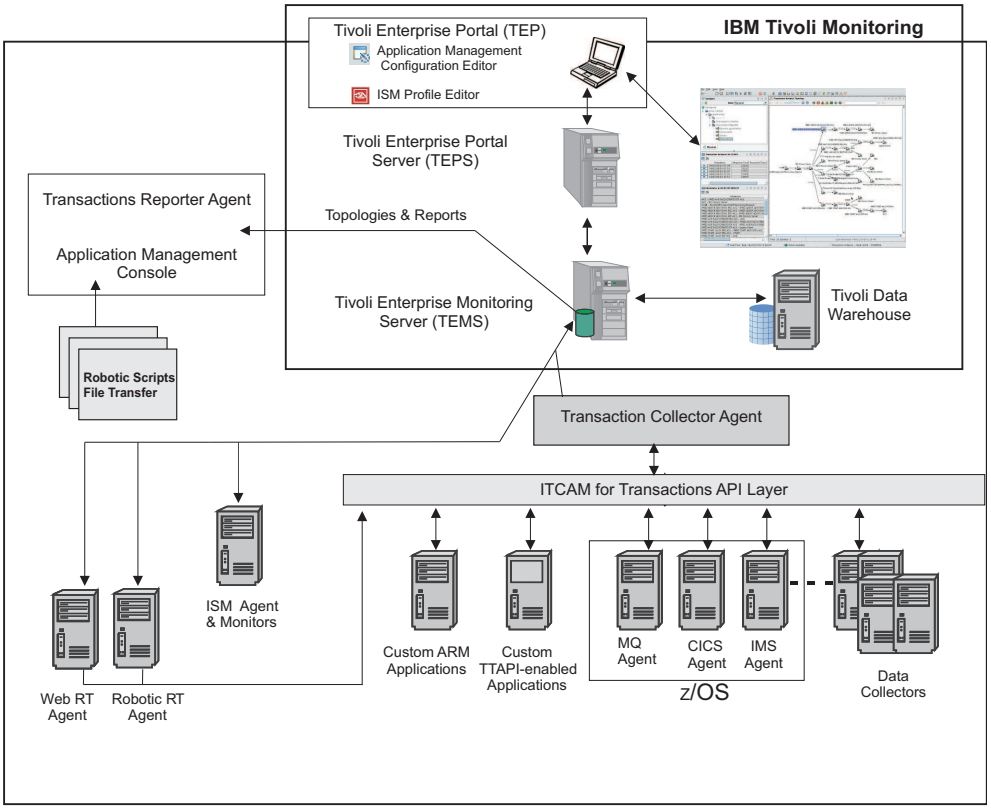


Figure 1. How ITCAM for Transactions integrates with IBM Tivoli Monitoring

Table 1 describes the main components illustrated in Figure 1.

Table 1. Tivoli Monitoring and ITCAM for Transactions integration

Component	Description
Tivoli Enterprise Portal Tivoli Enterprise Portal Server	The Tivoli Enterprise Portal Server enables retrieval, manipulation, and analysis of data from the agents. The server is between the client and the Tivoli Enterprise Monitoring Server (monitoring server). The Tivoli Enterprise Portal client is a Java-based user interface for viewing and monitoring your enterprise. It provides two modes of operation: desktop and browser. The Tivoli Enterprise Portal provides a consolidated view of the monitored environment so you can monitor and resolve performance issues. You can view your enterprise by using default physical views or by using custom created logical views in the Navigator.
Tivoli Enterprise Monitoring Server	Provides the collection and control point for alerts received from the monitoring agents and collects their performance and availability data. There are 2 types of monitoring servers: hub and remote.
Tivoli Data Warehouse	Stores historical data collected from monitoring agents. The data warehouse is located on a DB2®, Oracle, or Microsoft SQL database. To collect information to store in this database, you must install the Warehouse Proxy agent. To perform aggregation and pruning functions on the data, install the Warehouse Summarization and Pruning agent.

Table 1. Tivoli Monitoring and ITCAM for Transactions integration (continued)

Component	Description
Internet Service Monitoring agents and monitors	Provides the tools to identify availability and response time problems and monitors to test the availability and performance of various internet services, including monitoring websites, web-based e-commerce applications, and electronic mail (as well as the underlying services such as DNS, LDAP, and SMTP on which those services rely).
Response Time	Focuses on the end user experience, both real and simulated, by monitoring web transactions, playing back recorded scripts, and real user desktop experiences. Response Time includes the following components: • Application Management Console agent and Application Management Configuration Editor - enable you to define and configure the applications and transactions that you want to monitor. By applying common profile configurations across the environment, you can deploy monitoring in large-scale environments more efficiently. • Robotic Response Time - reports the results of simulated transactions (robotic scripts) so you can be proactive in managing availability and performance of your applications and identify bottlenecks before they impact customer satisfaction. • Web Response Time - reports real-user response time of web applications that can be broken down into browser (client) time, network time, server time, load time, and resolve time. Web Response Time monitors TCP traffic and detects components and protocols. It functions as an Aggregation agent for agentless tracking.
Transaction Tracking	Delivers an end-to-end view of your topology and response times across systems to quickly help isolate the cause of response time and availability problems. Transaction Tracking includes the following components: • Transaction Reporter - collects and stores the aggregated data from an Aggregation agent, such as the Transaction Collector and Web Response Time, and sends this data to the Tivoli Enterprise Portal workspaces. • Transaction Collectors - store the tracking data from multiple Data Collector plug-ins and compute aggregates. • Transaction Tracking API - is installed on each data collector and sends events and tracking information to Transaction Tracking. • Data Collector plug-ins - monitor traffic for specific applications and by using the Transaction Tracking API send this information to the Transaction Collectors. • Custom ARM applications - your own custom application that you can program to send events and provide tracking information to Transaction Tracking by using the Transaction Tracking API.
Aggregation agents	Agents that store the tracking data from monitors or Data Collector plug-ins, and compute aggregates for use by the Transaction Reporter. Aggregation agents include the Transaction Collector and Web Response Time (T5) agents.

For more information about how to use IBM Tivoli Monitoring and the Tivoli Enterprise Portal, see the publications available from IBM Tivoli Monitoring Information Center.

Integration with other products

Figure 2 shows how IBM Tivoli Monitoring and the monitoring agents integrate with other products.

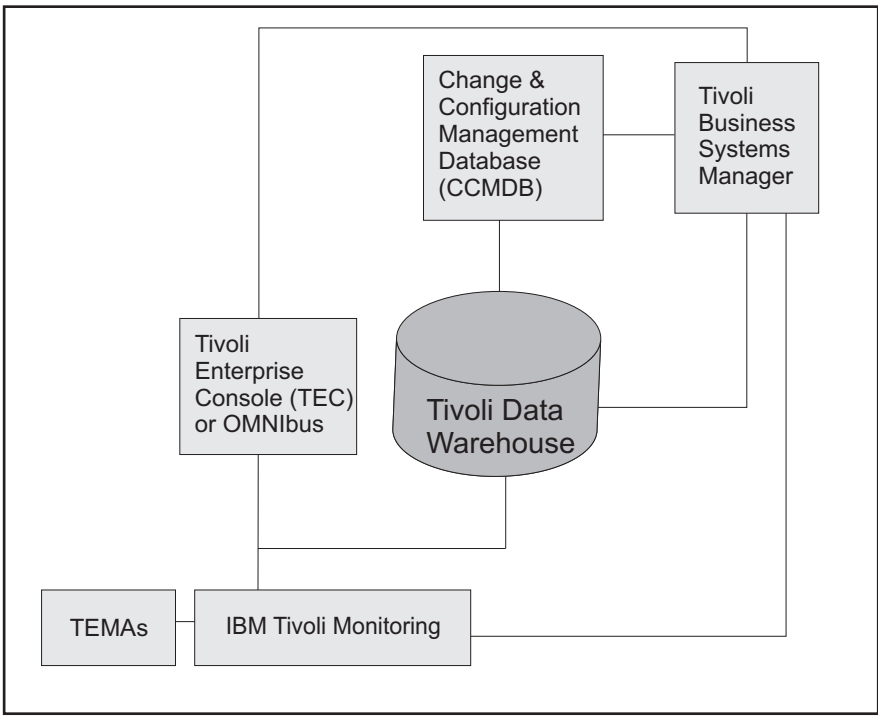


Figure 2. Integration of IBM Tivoli Monitoring and other products

Table 2 describes the components in Figure 2.

Table 2. How components integrate with IBM Tivoli Monitoring

Product	Description
Change and Configuration Management Database	Provides an enterprise-ready platform for discovering and storing deep, standardized data on configurations and change histories to help integrate people, processes, information, and technology.

Table 2. How components integrate with IBM Tivoli Monitoring (continued)

Product	Description
IBM Tivoli Business Systems Manager (TBSM) (Later versions renamed to IBM Tivoli Business Service Manager)	Manages real-time problems in the context of the business priorities for an enterprise. Business systems typically span web, client-server, or host environments and are made of many interconnected application components; they rely on diverse middleware, databases, and supporting platforms. TBSM provides customers a single point of management and control for real-time operations of end-to-end business systems management. You can graphically monitor and control interconnected business components and operating system resources from one single console and give a business context to management decisions. The software helps users manage business systems by understanding and managing the dependencies between business systems components and their underlying infrastructure. ITCAM for Transactions can be integrated with TBSM by using Omnibus. Situation events from Transaction Tracking can be forwarded from IBM Tivoli Monitoring to IBM IBM Tivoli Netcool/OMNIbus for display in TBSM. View these in TBSM by navigating to **Availability** > **Service Availability**. In the Service Tree, select **Imported Business Services** > **Transactions Business Activities** to display Transaction Tracking information. When you install Integration support by using the installation media provided with this release, you can access a new view of the data from Response Time and Transaction Tracking monitoring agents.
Tivoli Enterprise Management Agent (monitoring agents)	An IBM Tivoli Monitoring agent that is built on the IBM Tivoli Monitoring infrastructure. Tivoli Enterprise Management Agents connect to the Tivoli Enterprise Monitoring Server by using IPv4 or IPv6. Some configuration is required for IPv6. See the latest IBM Tivoli Monitoring Information Center for further information.
IBM Tivoli Monitoring (Tivoli Monitoring)	Provides monitoring for system level resources, detects bottlenecks and potential problems, and automatically recovers from critical situations to free system administrators from manually scanning extensive performance data during problem resolution. Upon notification of a poorly performing transaction component, you can launch either of the following products: • The *Tivoli Enterprise Portal* integrates and consolidates system monitoring end-to-end. The Tivoli Enterprise Portal provides a console from which you can monitor host and distributed systems. You can customize the information that you see in the Tivoli Enterprise Portal for your enterprise. See the IBM Tivoli Monitoring documentation for information about how to use the Tivoli Enterprise Portal. • *Tivoli Data Warehouse* enables you to drill down to a lower level of a transactions and historical data, and enables you to identify issues such as poorly configured systems. With the addition of products such as IBM Tivoli Monitoring for Databases, IBM Tivoli Monitoring for Web Infrastructure, and IBM Tivoli Monitoring for Business Integration, you can further diagnose infrastructure problems and, in many cases, resolve them before they affect the performance of business transactions.

About Internet Service Monitoring

The information gathered and processed by Internet Service Monitoring enables you to determine whether a particular service is performing adequately, identify problem areas, report service performance measured against Service Level Agreements (SLAs), and forward performance data to IBM Tivoli Monitoring, IBM Tivoli Composite Application Manager for Transactions, and other event management tools such as IBM Tivoli Netcool/OMNIbus.

Internet Service Monitoring works by emulating the actions of a real user. For example, the HTTP monitor tries to access particular web pages, then measures how well the HTTP service performed. The data recorded by the monitor provides an immediate indication of the status of the HTTP service to the service operators, and can also be used to provide reports on service performance.

Internet Service Monitoring architecture

The core components of the Internet Service Monitoring architecture are the Internet service monitors.

The Internet service monitors regularly poll or test Internet services to check their status. The test results generate data for SLA evaluation, reporting, and alert generation. Internet Service Monitoring can monitor the protocols listed in Table 3.

Table 3. List of protocols monitored by Internet Service Monitoring

Protocols Internet Service Monitoring monitors			
DHCP	ICMP	RADIUS	SNMP
Dial - deprecated in ITCAM for Transactions V7.3	IMAP4	RPING	SOAP
DNS	LDAP	RTSP	TCPPort
FTP	NNTP	SAA	TFTP
HTTP	NTP	SIP	WMS - deprecated in ITCAM for Transactions V7.3
HTTPS	POP3	SMTP	Combinations of the other protocols by using TRANSX

Figure 3 on page 9 shows a typical Internet Service Monitoring deployment.

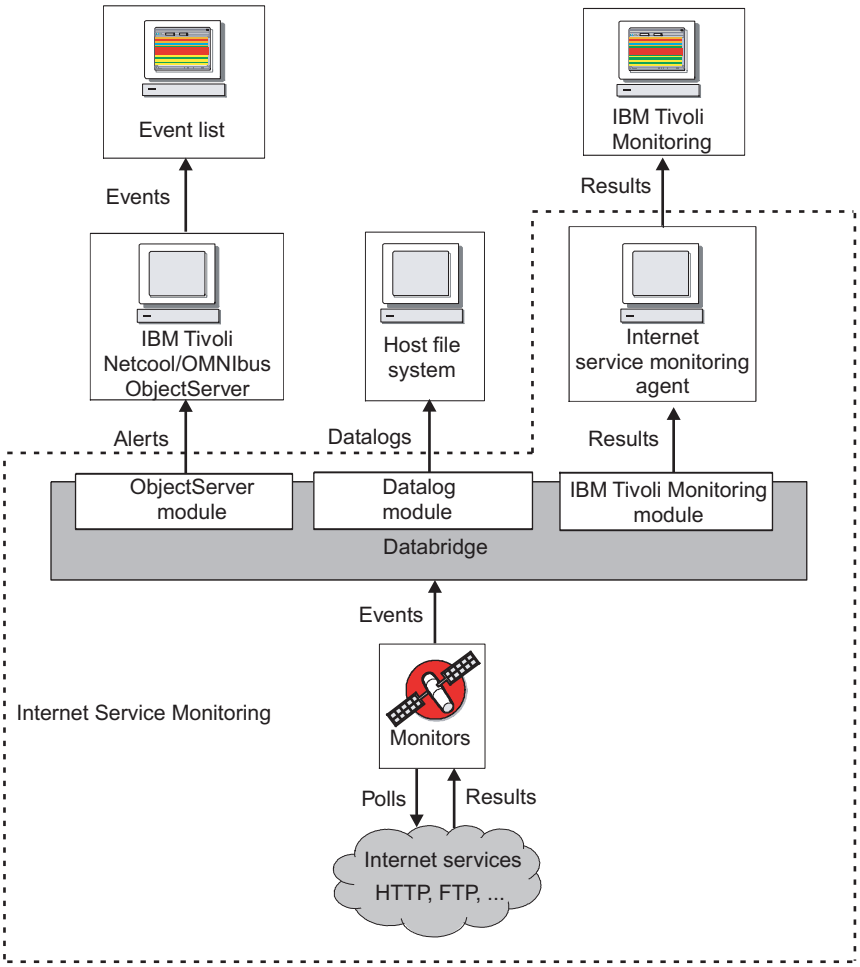


Figure 3. Internet Service Monitoring architecture

Figure 3 shows the following Internet Service Monitoring components:

Monitors

Test the specific Internet services and forward the test results to the Databridge. They emulate the actions of a real user of the service. For example, the HTTP monitor periodically attempts to access a web page by emulating requests that a web browser would usually send when a user visits the page. It generates an event containing the results of the test (including status information) which is sent to the Databridge.

Monitors are distinguished from IBM Tivoli Netcool/OMNIbus probes by their polling functions. Probes connect to an event source to acquire the event data that it generates, while monitors actively poll or test services at regular intervals by injecting transactions or queries into the target service, and generating performance evaluation data.

Databridge

Acts as the communications bridge between the monitors, the IBM Tivoli Netcool/OMNIbus ObjectServer, and the Internet service monitoring agent. The Databridge receives the results of service tests performed by the monitors and converts this data into different formats for processing by the ObjectServer and the monitoring agent. The Databridge can also generate

XML datalogs that you can use for archiving or simple reporting purposes. Detailed reporting is available within IBM Tivoli Monitoring through workspaces.

Internet service monitoring agent
Converts test results into the format required by IBM Tivoli Monitoring.

ObjectServer module
Converts events into alerts containing SLA and performance data and sends these alerts to the IBM Tivoli Netcool/OMNIbus ObjectServer. IBM Tivoli Netcool/OMNIbus users can then view service status information in the Event List. IBM Tivoli Netcool/OMNIbus ObjectServer and the Event List are part of IBM Tivoli Netcool/OMNIbus and are not installed with Internet Service Monitoring.

Datalog module
Converts test results to XML and then sends this information to a host file system for archiving or simple reporting purposes. The XML is useful for customers who have developed their own reporting tools and want to continue working with these tools.

IBM Tivoli Monitoring module
Sends results to the Internet service monitoring agent that uses a mapping file to convert the results into the format required by IBM Tivoli Monitoring for reporting in workspaces.

About Response Time

The Response Time component of ITCAM for Transactions provides a targeted solution for managing composite applications. It is designed to provide support staff with the information they need to assess whether composite applications are working correctly everywhere in the network. This functionality plays a dual role in enterprise IT. If a composite application is used within your own enterprise environment, you might be able to tolerate a slight drop in performance that has little or no effect on your financial results. If, however, a composite application is used by external customers, a drop in performance might have legal consequences due to violations of preestablished Service Level Agreements (SLAs). While neither of these scenarios is desirable, both are addressed, and in many cases precluded, by the monitoring capabilities provided with Response Time agents. The software offers the following features:

- A single infrastructure built on IBM Tivoli Monitoring.
- A consolidated user interface built on Tivoli Enterprise Portal (TEP), which offers single sign-on and common reporting.
- The ability to fully customize the reports and workspaces.
- Intelligent alerts based on IBM Tivoli Monitoring situations.
- Reports and alerts for real-time or historical metrics.
- Identifies bottlenecks in the Client, Network, or Server (CNS) by breaking down response time data into segments so that you can understand trends and system loads.
- Identifies, reports, and sends alerts on individual clients or locations.
- Discovers, reports on, and sends alerts for backend server resources.
- Provides the capability for configuring data aggregation as frequently as every 5 minutes.
- Provides simplified configuration, including default situations.

Response Time includes the following monitoring agents:

- Application Management Console
- Robotic Response Time
- Web Response Time

Application Management Console

Application Management Console provides an accurate snapshot of ITCAM for Transactions monitoring in near real time. It provides real-time aggregated and consolidated application and transaction availability and response time information for all applications monitored by Internet Services, Response Time, and Transaction Tracking monitoring agents. It collects data in real time at a configurable, constant interval instead of relying on the Tivoli Data Warehouse.

Use the Application Management Console to see status summary and trend analysis information across managed resources and to perform problem determination. This information is displayed in the Tivoli Enterprise Portal.

The Application Management Console agent is required when other ITCAM for Transactions agents are installed. The Application Management Console agent manages and distributes profiles, maintenance windows, client information, and user information for all the other Response Time and Transaction Tracking monitoring agents.

Robotic Response Time

Robotic Response Time provides active monitoring of customer business transactions. These business transactions represent a complex set of steps typically performed by an end user to complete a business objective, such as logging in to an online banking application, checking an account balance, and transferring funds. This set of steps can be recorded and played back by using this agent to verify availability and performance. It is installed separately on various desktop and server systems in your enterprise and on the internet.

Monitoring can be completed from the start of a transaction and, because it is enabled to support TTAPI and can be integrated into the Transaction Collector and Transaction Reporter functions, you can display end-to-end topology views of your robotic transactions as they flow through the system.

Robotic Response Time provides the following features:

- Improved robotic monitoring with Rational Performance Tester.
- Playback of scripts by using Rational Functional Tester against Windows applications, including 3270 applications.
- Immediate playback of robotic scripts.
- Monitoring causes of script failure by viewing actual screen captures and HTML data captures from failed playback sessions in Rational Performance Tester and Rational Functional Tester.
- Monitoring the performance and availability of applications to detect problems before end-users experience them. Robotic Response Time performs this monitoring by using robotic technology to record and play back transactions to determine if the transaction is performing as expected.

Robotic Response Time provides robotic monitoring for the following applications:

- Web applications that use HTTP and HTTPS protocols
- Microsoft Windows GUI client applications

- Applications or scripts with a command-line interface, such as:
  - Custom monitoring scripts
  - Applications such as DB2 that provide a command-line interface
  - Playback technologies such as Rational Functional Tester or `wget`
- Mercury LoadRunner HTTP and HTTPS scripts
- Citrix hosted applications
- SAP
- Siebel
- Web Services
- Oracle ERP Applications
- Robotic scripts file transfer - discovers and uploads all of the files and file dependencies that are required for robotic scripts. You can also instruct the tool to automatically ARM-instrument a recording that has not previously been instrumented. Robotic scripts record a sequence of steps in a transaction to simulate a particular business transaction executed from specific locations so you can monitor end-user experience with Robotic Response Time

See the *Administrators Guide* for more information about using Rational Performance Tester and Rational Functional Tester with Robotic Response Time.

Web Response Time

Web Response Time provides real end-user monitoring of client web requests to server components. It can be installed locally on the server system, or on a separate system. Web Response Time uses server-side monitoring to capture HTTP and HTTPS transaction data such as response time and status codes. You can use it to capture the performance and availability data of actual users for Service Level Agreement (SLA) reporting. Web Response Time also detects protocols and applications by monitoring TCP/IP network flows.

Using Web Response Time, you can perform the following tasks:

- Monitor end-user performance and availability for web-based applications.
- Capture web request response time and its segmentation.
- Monitor the performance of web page request and each embedded object in that web page. This feature, which can be switched on or off, can identify if any graphics, tables, JavaScript, or Applets are causing response time problems. Audio or video request monitoring is not available.
- Capture and report on HTTP query string and FORM Post data.
- Monitor response times, including response time of the workstation, without being physically located on the workstation.
- Monitor HTTP and HTTPS transactions while running in Appliance Mode.

  If you prefer to not modify your web server, you can install the Web Response Time agent in appliance mode, either locally on the server, or remotely on a different host that utilizes a network tap, port spanning, or a hub to gain access to the network traffic of the server. With this configuration, you can monitor your web servers without modifying or impacting the server systems. This method is the preferred method of installation.
- Monitor specific users by their sessions and user names.
- By default, the Web Response Time agent monitors all network interfaces. However, it can also monitor a specific network interface. By default, the Web

Response Time Analyzer automatically selects the correct interface. However, you can limit it to one network interface.

- Monitor transactions from web servers to WebSphere Application Server by using the Web Response Time - Transaction Tracking integration option.

The Web Response Time agent can track transactions without the need for domain-specific or application-specific data collectors. This type of monitoring is called *agentless transaction tracking*, and extends the capabilities of existing ITCAM for Transactions features and functions:

- Monitors generic TCP/IP based network flows.
- Enhanced Tivoli Enterprise Portal workspaces provide additional capabilities to visualize network flow data and dependencies.
- The Transaction Reporter agent uses data from the Web Response Time agent along with data from existing domain-based data collectors, such as Data Collector for WebSphere Message Broker, to display this TCP/IP data in topology views. These topology views can include data from both traditional agent-based data sources and agentless tracking data sources. Using these capabilities together, you can deploy Web Response Time agents to collect data, then display the resulting topology, and successively deploy agent-based data collectors to obtain more detailed tracking information.
- You can use additional capabilities in the Application Management Configuration Editor to create and modify configurations that the Web Response Time agent applies to the monitored TCP/IP network flow data.

Web Response Time is also an Aggregation agent.

Aggregation agents are monitoring agents that provide data storage and compute aggregates for Transaction Tracking. Aggregation agents include Transaction Collectors and Web Response Time agents.

Aggregation agents, including Transaction Collectors and Web Response Time, communicate with the Transaction Reporter through the Tivoli Enterprise Monitoring Server. Multiple Aggregation agents can report to a single Transaction Reporter. Each Aggregation agent can be queried by one or more Transaction Reporters. Transaction Collectors do not communicate with each other.

About Transaction Tracking

Transaction Tracking traces transactions within and between applications. It determines the time spent by the transaction in each application and, where possible, the time spent communicating between applications. It can then generate alerts based on thresholds which specify minimum or maximum permissible values for specific attributes.

Transaction Tracking accommodates a range of different products, correlation techniques, and transaction topologies. It enables you to expand the capabilities of services, and can be customized for specific environments. Transaction Tracking also tracks individual transactions where the correlation techniques make this possible.

Within a complex transaction topology, the transaction path cannot always be determined because of the differences in correlation techniques used. Therefore, as the information from a transaction becomes available, Transaction Tracking uses this accumulated information to start determining the type and details of the full transaction.

The main components of Transaction Tracking are:

- Data Collector plug-in

  Data Collector plug-ins are a combination of Transaction Tracking API and supporting files that are installed on a *domain*. The Data Collector plug-in enables an application to transmit tracking data to a Transaction Collector.

- Aggregation agents

  Aggregation agents are monitoring agents that provide data storage and compute aggregates for Transaction Tracking. Aggregation agents include Transaction Collectors and Web Response Time agents.

  Transaction Collectors provide distributed storage for all *instance data* that is collected from multiple *data sources*. Transaction Collectors also compute *aggregates*. Configure a Transaction Collector by using the Manage Tivoli Enterprise Monitoring Services console, or remotely in the Tivoli Enterprise Portal if the IBM Tivoli Monitoring operating system agent is installed in the same home directory and is connected to the same IBM Tivoli Monitoring system.

  Aggregation agents, including Transaction Collectors and Web Response Time, communicate with the Transaction Reporter through the Tivoli Enterprise Monitoring Server. Multiple Aggregation agents can report to a single Transaction Reporter. Each Aggregation agent can be queried by one or more Transaction Reporters. Transaction Collectors do not communicate with each other.

- Transaction Reporter

  The Transaction Reporter is an IBM Tivoli Monitoring Tivoli Enterprise Management Agent (TEMA). The Transaction Reporter contains a number of algorithms that create transaction topologies or transaction instance graphs which are displayed in the Tivoli Enterprise Portal. A single Transaction Reporter can receive information from one or more Transaction Collectors through the Tivoli Enterprise Monitoring Server. Configure the Transaction Reporter by using the Manage Tivoli Enterprise Monitoring Services console, or remotely in the Tivoli Enterprise Portal if the IBM Tivoli Monitoring operating system agent is installed in the same home directory and is connected to the same IBM Tivoli Monitoring

- Predefined content

  Predefined content includes workspaces, situations, and Take Action commands. The workspaces contain charts or tables showing aggregated data, and are divided into four conceptual monitoring models: applications, components, servers, and transactions. Situations are tests that check the aggregated data against a set of conditions and take action when the conditions are met.

  You can modify the predefined content to create workspaces specific to your organization if required.

IBM Tivoli Composite Application Manager for Transactions also uses the following:

- Tivoli Data Warehouse

  The Tivoli Data Warehouse stores long term historical data.

- Application Management Console

  Application Management Console provides a default set of configuration mappings to Application Response Measurement-enabled applications such as WebSphere® Application Server and DB2, and to applications such as CICS®, IMS™, IMS Connect, ITCAM for SOA, and WebSphere Application Server by using ITCAM for Application Diagnostics. By using these configuration

mappings, you can track some transactions automatically without further configuration. These mappings are displayed in the Application Management Configuration Editor.

Application Management Configuration Editor is installed with the Application Management Console (t3 agent) and is shared with the Response Time component.

How IBM Tivoli Composite Application Manager for Transactions fits into IBM Tivoli Monitoring

IBM Tivoli Composite Application Manager for Transactions integrates with the IBM Tivoli Monitoring framework. It provides enhancements to the existing infrastructure and new components.

Design

IBM Tivoli Composite Application Manager for Transactions integrates with the IBM Tivoli Monitoring framework by using the Tivoli Enterprise Portal, Tivoli Data Warehouse, situations, and workspaces to collect and display information about transaction response times and interactions. You can access workspaces through the Tivoli Enterprise Portal. The Tivoli Enterprise Portal communicates with the Tivoli Enterprise Portal Server and the Tivoli Enterprise Monitoring Server that form part of the standard IBM Tivoli Monitoring framework.

Figure 4 on page 16 illustrates this design.

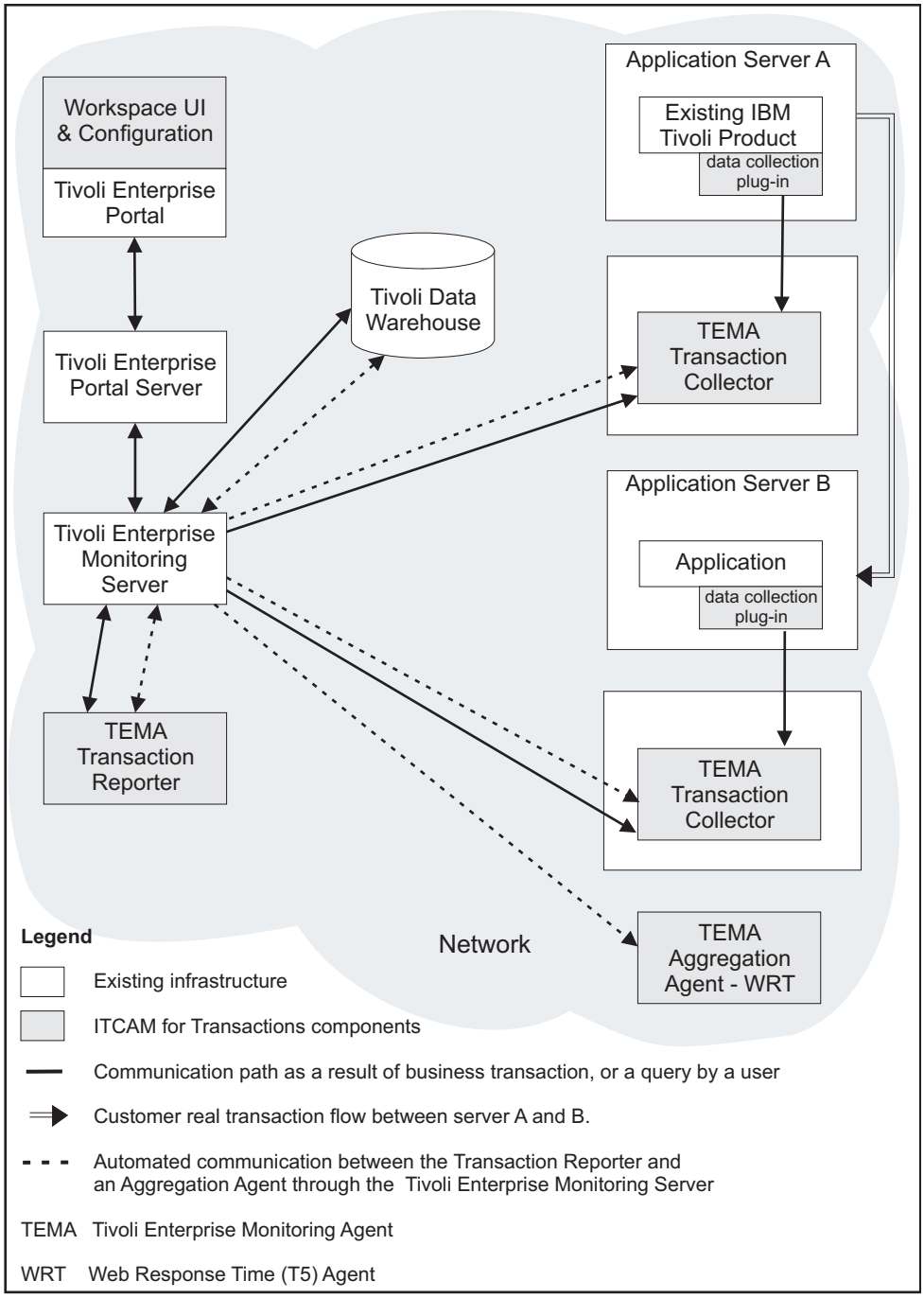


Figure 4. How Transaction Tracking fits in to IBM Tivoli Monitoring

Figure 4 displays how IBM Tivoli Composite Application Manager for Transactions fits into the IBM Tivoli Monitoring framework. As you request information in the Tivoli Enterprise Portal, a series of events are triggered throughout the framework, which are indicated by solid lines in the diagram.

The dotted lines show the communication paths between the Transaction Reporter and Aggregation agents, such as the Transaction Collector, through the Tivoli Enterprise Monitoring Server. This is an automatic process that happens in the background at configurable intervals, and is not necessarily initiated by user

requests. For further information on configuring the collection time interval, see *Data collection* in *ITCAM for Transactions Administrator's Guide*.

The Transaction Reporter communicates with the Transaction Collector through the Tivoli Enterprise Monitoring Server to obtain *aggregate* and *instance* data, and uses this data to display the transaction topology at various levels of detail:

- **Summary** workspaces provide an overall view of applications communicating with other applications.
- **Interaction Detail** workspaces and **Transaction Instance** workspaces provide a specific view of interactions for a transaction instance.
- **Topology** workspaces display the aggregate topology or a specific instance topology.

The Transaction Reporter also communicates with other Aggregation agents, such as Web Response Time, through the Tivoli Enterprise Monitoring Server to obtain network information and uses this data to display data and topologies in **Transactions Overview** and **Agentless Data** workspaces.

The Tivoli Enterprise Monitoring Server provides a communication mechanism only, it does not store any data. Data is stored in the Tivoli Data Warehouse for days, in the Transaction Reporter for hours, and in the Transaction Collector for minutes. For further information on configuring the collection time interval, see *Data collection* in *ITCAM for Transactions Administrator's Guide*.

The Transaction Reporter provides Instance Data only to the Tivoli Data Warehouse via the Warehouse Proxy for Instances that have been requested by a Take Action command. This action is performed by the *Slow_Transaction* situation. Viewing Instance data in the **Transaction Instance** workspace or **Instance Topology** does not make this data available to the Warehouse Proxy.

The Transaction Collector and other Aggregation agents are IBM Tivoli Monitoring Tivoli Enterprise Management Agents. Both the Transaction Reporter and Transaction Collector agents are deployed and configured by using the installer and can be reconfigured by using the Manage Tivoli Enterprise Monitoring Services console.

The Transaction Collector does not provide any data directly to the Warehouse Proxy.

IBM Tivoli Composite Application Manager for Transactions integrates with other products in the Tivoli Enterprise Portal, and you can launch a workspace for another product from an IBM Tivoli Composite Application Manager for Transactions workspace.

How IBM Tivoli Composite Application Manager for Transactions works

IBM Tivoli Composite Application Manager for Transactions provides new components that fit into IBM Tivoli Monitoring and interact with each other to provide views of transaction response times and interactions.

The main components in IBM Tivoli Composite Application Manager for Transactions are the Data Collector plug-ins (including the Transaction Tracking API), Transaction Collectors, Transaction Reporter, and the workspaces displayed in the Tivoli Enterprise Portal.

Figure 5 shows how the components within IBM Tivoli Composite Application Manager for Transactions interact. It shows how Data Collector plug-ins send data to their associated Transaction Collector, and that the Transaction Reporter obtains data from various Transaction Collectors and other Aggregation agents. It also shows how applications can interact with each other even though the data is sent to different Transaction Collectors.

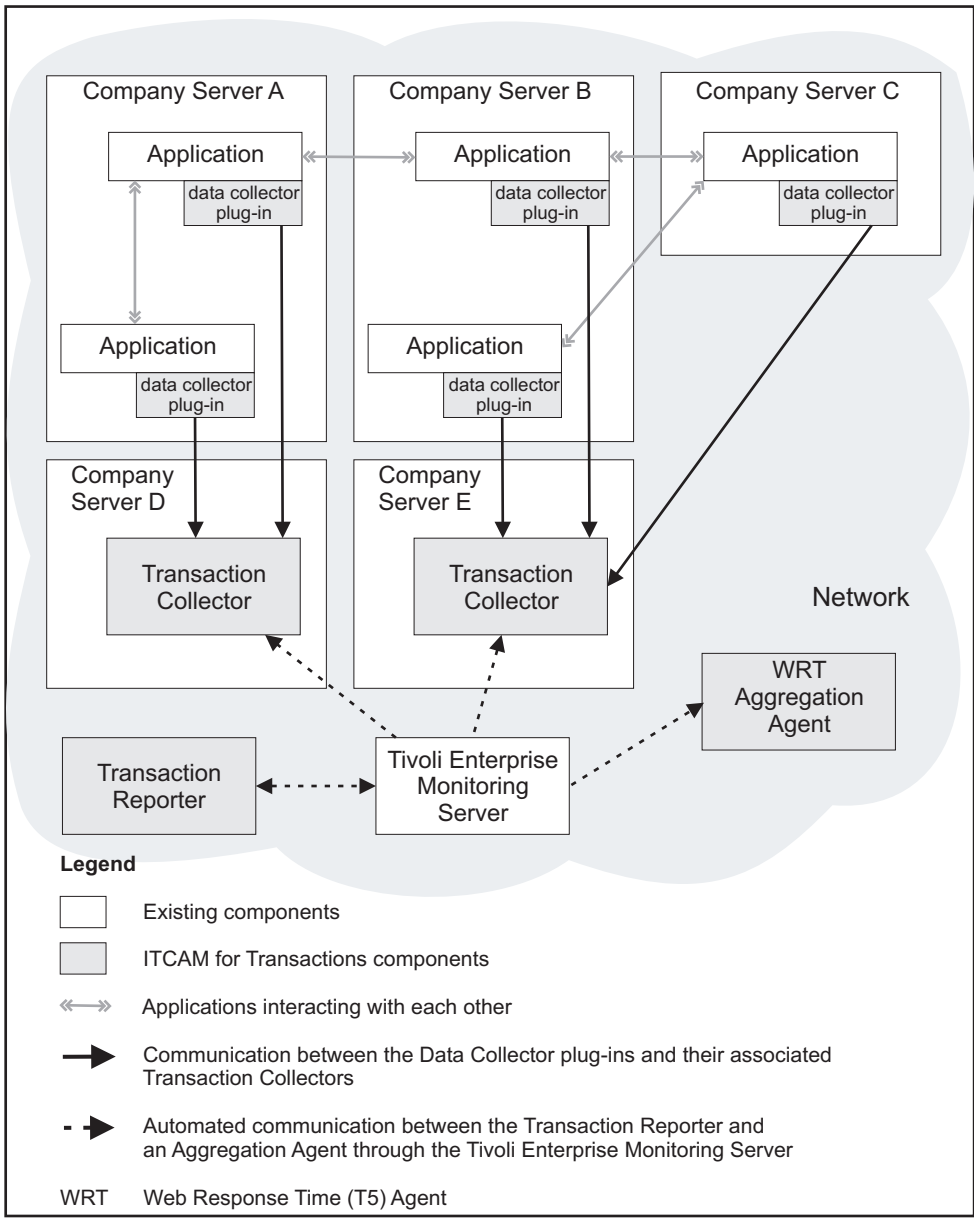


Figure 5. IBM Tivoli Composite Application Manager for Transactions component interaction diagram

Data collector plug-ins

Data Collector plug-ins monitor specific applications. They encode application data and transfer it to a Transaction Collector by using the Transaction Tracking API. Data Collector plug-ins are located on the same server as the application they

serve. The data gathered by Data Collector plug-ins is used to build comprehensive topologies and display information about transaction response times and interactions.

Different applications can communicate with each other, but each application has its own Data Collector plug-in and transfers data to a Transaction Collector.

Table 4. Transaction Tracking Data Collector plug-ins

Data Collector plug-in	Description
CICS Tracking	An extension to Transaction Tracking for z/OS® that provides IBM CICS support on the z/OS operating system. The CICS agent automatically tracks External Call Interface (ECI), Dynamic Program Link (DPL), IBM MQSeries®, and SOAP over HTTP traffic and uses the Transactions Base to send events to a Transaction Collector. You can also send your own events from CICS exits or applications by using the Transactions Base API or send Transactions events by using the provided CICS program.
CICS TG Transaction Tracking	Tracks interactions between applications that pass through CICS Transaction Gateway environments, enabling you to monitor the performance of CICS Transaction Gateway components and their effect on your enterprise's applications. Use CICS TG Transaction Tracking with ITCAM for Application Diagnostics and CICS Tracking for complete correlation of transactions flowing from WebSphere Application Server through the CICS TG Gateway Daemon into CICS.
CICS TXSeries Data Collector	Integrates with CICS TXSeries for AIX and collects data related toCICS TXSeries transactions and programs.
IMS Tracking	An extension to Transaction Tracking for z/OS that provides IBM IMS support on the z/OS operating system.
MQ Tracking	An extension to Transaction Tracking for z/OS that provides support for WebSphere MQ on z/OS and distributed operating systems. The MQ agent tracks MQ events and forwards them to a Transactions Collector.
Tuxedo Tracking	Tracks transactions between applications in the Tuxedo application and monitors the performance of these interactions.
WASTT	Tracks interactions between ARM-instrumented applications on WebSphere Application Server and other domains, such as WebSphere MQ.
Data Collector for WebSphere Message Broker	Tracks interactions between applications that pass through WebSphere Message Broker environments. Data Collector for WebSphere Message Broker uses the KK3UserExit WebSphere Message Broker user exit to collect the data for tracking transactions. After analyzing the data, the KK3UserExit user exit dispatches the data as transaction tracking events to a Transaction Collector.
ITCAM for SOA	ITCAM for SOA integrates with ITCAM for Transactions and displays information in Transaction Tracking workspaces and views.

Table 4. Transaction Tracking Data Collector plug-ins (continued)

Data Collector plug-in	Description
Custom ARM Applications	An application that already contains the necessary ARM function calls. You can monitor generic ARM applications (such as the web server plug-in for IBM WebSphere Application Server, IBM WebSphere, or IBM DB2) with Transaction Tracking agents.
Custom User Applications	Your own custom application that you can program to send events and provide tracking information to Transaction Tracking by using the Transaction Tracking API.

Aggregation agents

Aggregation agents are monitoring agents that provide data storage and compute aggregates for Transaction Tracking. Aggregation agents include Transaction Collectors and Web Response Time agents.

Transaction Collectors receive instance data from applications through the Transaction Tracking API installed with Data Collector plug-ins. Install the Transaction Collector on a separate server to the applications. The server should not have critical applications running on it, and have sufficient resources available to run the Transaction Collector.

The Transaction Collector stores this data, computes aggregates, and responds to queries for data from the Transaction Reporter through the Tivoli Enterprise Monitoring Server. Multiple Transaction Collectors can provide data to a Transaction Reporter, but Transaction Collectors do not communicate with each other.

A Transaction Collector removes data older than a configurable age, or because it has reached a configurable volume. For further information on configuring the collection time interval, see "Tuning data collection" in the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide*.

Transaction Reporters

The IBM Tivoli Composite Application Manager for Transactions workspaces use data from the Transaction Reporter. The Tivoli Enterprise Monitoring Server enables the Transaction Reporter to query one or more Aggregation agents for aggregate data. This happens in the background at set intervals. For further information on configuring the collection time interval, see *Tuning data collection* in the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide*.

After receiving and caching the aggregate data, the Transaction Reporter collects a subset of instance data. It then uses algorithms to build transaction topologies that are applied to the aggregate data to produce interaction data. The aggregate data and interaction data are displayed in the workspaces and provide an overview of the transaction performance, including the alternate paths a transaction may take. The Transaction Reporter may need to contact the Aggregation agents multiple times to obtain enough tracking data to create a complete transaction topology.

The Transaction Reporter's interaction views at an Aggregate Level, not the Instance level, are an estimate based on the individual Aggregates supplied by the Transaction Collectors and a Topology determined by the Transaction Reporter.

This Topology is determined by sampling the Transaction Collector for some Instance data, then performing tracking for several hops, and then identifying an Aggregate from the Context information in the Instance Data.

When determining Aggregate Interaction Rows, the Transaction Reporter may receive individual Aggregates that have differing counts because the Transaction Collector determines which time period to update, based on the time stamp of the initial instance event for a specific transaction.

For example, if an Instance Level interaction occurs from *A* to *B*, it is possible for *A*'s Aggregate to be in one period and *B*'s Aggregate to be in the following period. The higher the interaction rate, the less significant any difference between the count in Aggregates *A* and *B* will be. However, as the transaction rate approaches 0 it may be that no interactions are determined, as an Aggregate for *A* occurs in one time period, and an Aggregate for *B* occurs in another. Moving from Aggregates to Instance Interactions would display a topology of *A* to *B*, but the Aggregate Interactions Topology would show only *A* or *B*.

Note: In ITCAM for Transactions V7.2.0.1 and later, when the Transaction Reporter queries the Transaction Collector for a single instance, the Transaction Reporter now traces only that single instance.

The Transaction Reporter also provides data that enables workspaces to display specific instance graphs that provide the exact set of interactions that occurred during the processing of a single transaction instance. Historical instance information can also be displayed, see Transactions: Historical Transaction Instances in the *User's Guide* for further information.

Use the Manage Tivoli Enterprise Monitoring Services console to link the Transaction Reporter to specific Transaction Collectors. The default setting is for the Transaction Reporter to collect data from every Transaction Collector available through the Tivoli Enterprise Monitoring Server.

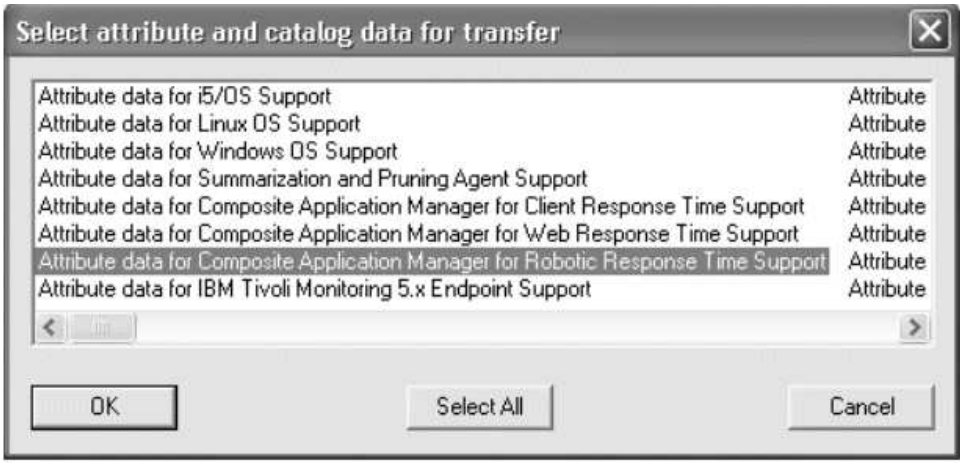
Chapter 2. Tivoli Enterprise Monitoring Server support on z/OS

If any ITCAM for Transactions agents are connected to a Tivoli Enterprise Monitoring Server running on z/OS, you must upload the corresponding catalog and attribute data files and install application support.

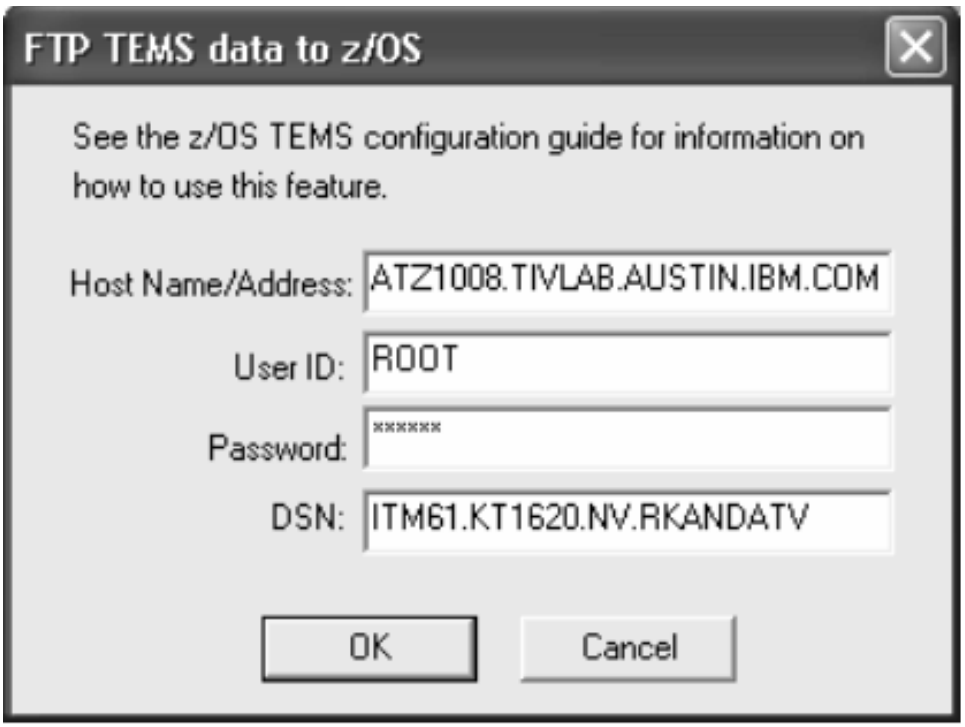
Uploading catalog and attribute data files

To enable the ITCAM for Transactions agents to run against a Tivoli Enterprise Monitoring Server on z/OS, you must install the catalog and attribute data files on the hub Tivoli Enterprise Monitoring Server. Follow these steps:

1. Open the Manage Tivoli Enterprise Monitoring Services application.
2. Select **Advanced** > **Utilities** > **FTP Catalog and Attribute files**.
3. In the **Select attribute and catalog data** window, select all the agents that apply and click **OK**.



4. In the **FTP TEMS Data to z/OS** window, enter the following information and click **OK**:
   - The fully qualified host name of the hub Tivoli Enterprise Monitoring Server.
   - A valid FTP user ID and password.
   - The fully qualified name of the `RKANDATV` data set.

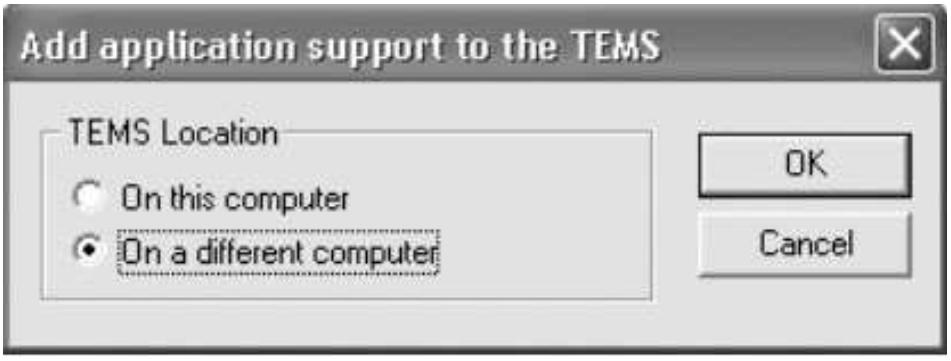


5. Click **OK** again in the confirmation window.

Adding Response Time support for Tivoli Enterprise Monitoring Server on z/OS

The Tivoli Enterprise Portal and Tivoli Enterprise Monitoring Server must be running to add application support for the Tivoli Enterprise Monitoring Server.

1. From the Manage Tivoli Enterprise Monitoring Services window, verify that the Tivoli Enterprise Portal is started.
2. From the Manage Tivoli Enterprise Monitoring Services window, right-click Tivoli Enterprise Portal Server.
3. Select the **Actions** from drop-down menu.
4. Select **Advanced Add application support to the TEMS**.
5. Select **On a different computer** from the Add Application Support to the TEMS window.

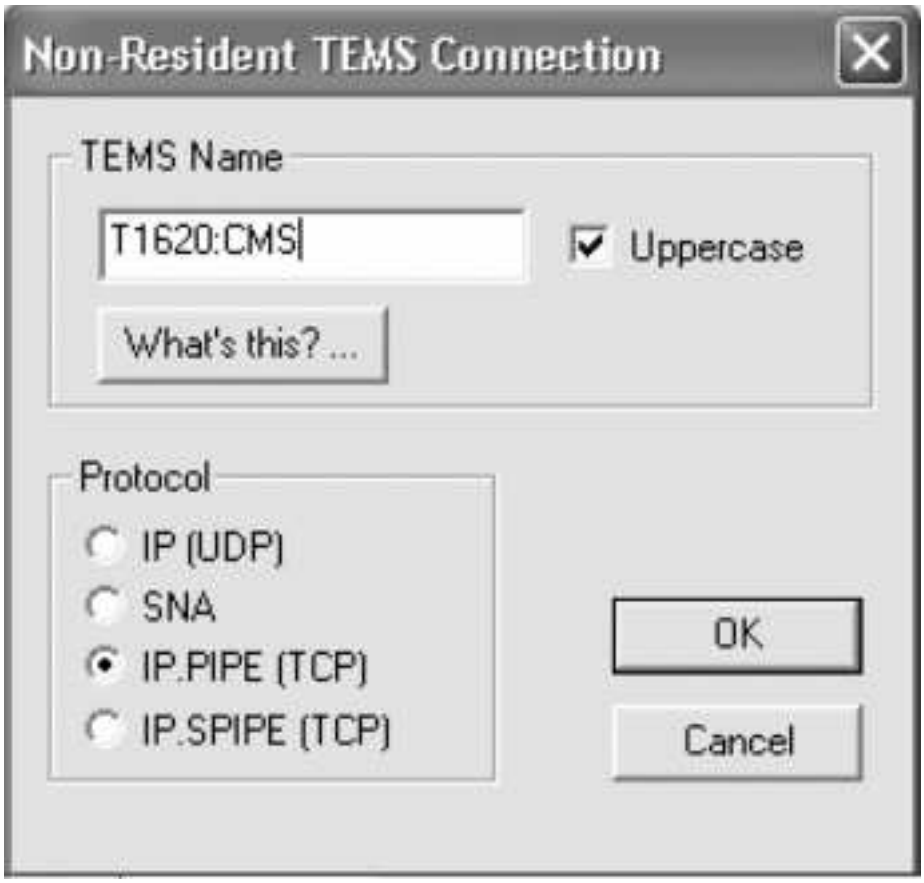


6. The software displays a confirmation window.

Complete one of the following steps:

- If the Tivoli Enterprise Monitoring Server is started, click **OK** and go to the next step.
- If the Tivoli Enterprise Monitoring Server is not started, start it, as explained in "Starting the hub Tivoli Enterprise Monitoring Server on z/OS" on page 30, and click **OK**.

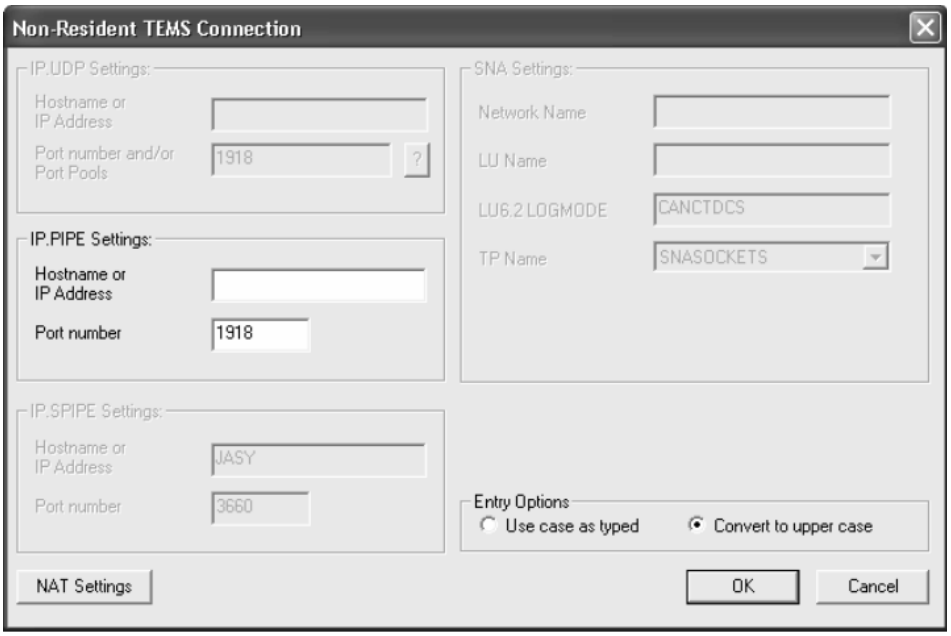
7. Provide the communication protocol and TEMS Node ID for the Non-resident TEMS Connection window and click **OK**.



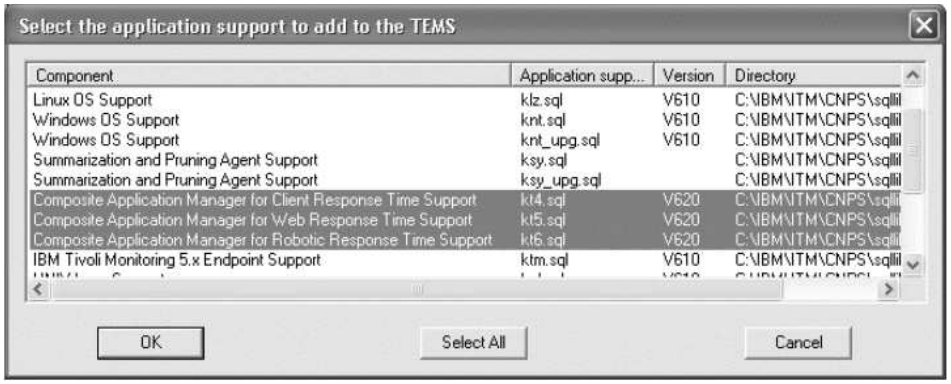
- You can locate the value for the CMS Node ID for the hub Tivoli Enterprise Monitoring Server you configured by examining the `CMS_NODEID` parameter in `KDSENV` member in the `&rhilev.&midlev.RKANPARU` data set on the z/OS system where you configured the Tivoli Enterprise Monitoring Server hub.

- You must select a communication protocol that the Tivoli Enterprise Monitoring Server was configured to support in your environment.

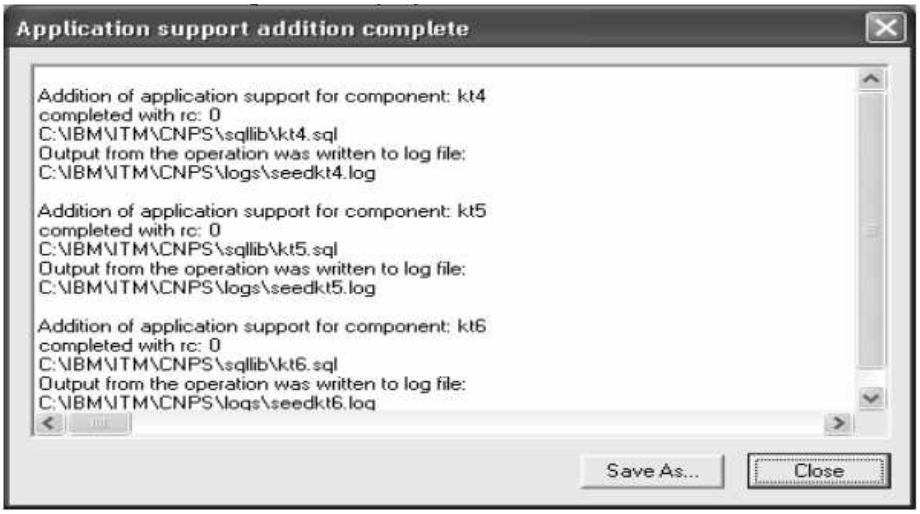
8. Provide the following information for the Non-resident TEMS Connection settings window and click **OK**.
   - **Hostname or IP Address** is the fully qualified hostname or IP address of the Tivoli Enterprise Monitoring Server.
   - **Port #** is the port number that Tivoli Enterprise Portal and Tivoli Enterprise Portal Server uses to communicate with theTivoli Enterprise Monitoring Server on z/OS. The default is 1918.
   - **Entry Options** is the case in which all entries on this screen are processed.



9. Select all the ITCAM for Transactions agents from the Select the Application Support to Add to the TEMS window and click **OK**.



10. Do one of the following:
    - If you see the following confirmation window, adding application support is completed successfully. Go to "Starting the hub Tivoli Enterprise Monitoring Server on z/OS" on page 30.

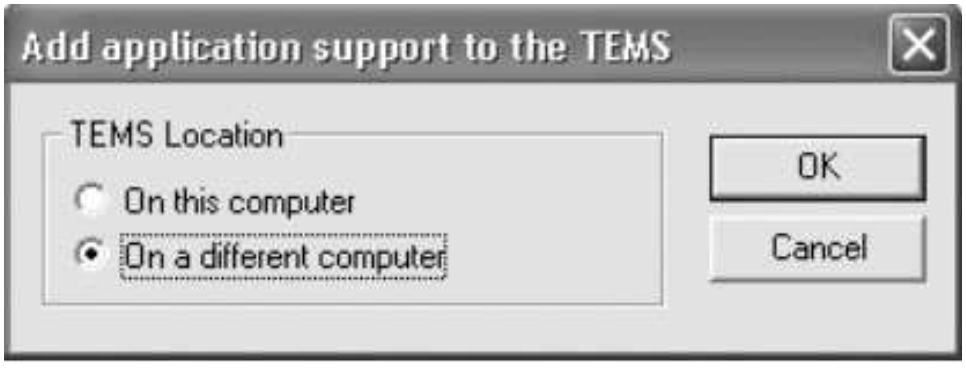


- If you do not see this window, look in the `IBM\ITM\CNPS\Logs` file for diagnostic messages to determine the cause of the problem.

Adding Transaction Tracking support for Tivoli Enterprise Monitoring Server on z/OS

The Tivoli Enterprise Portal and Tivoli Enterprise Monitoring Server must be running to add application support for the Tivoli Enterprise Monitoring Server.

1. From the Manage Tivoli Enterprise Monitoring Services window, verify that the Tivoli Enterprise Portal is started.
2. From the Manage Tivoli Enterprise Monitoring Services window, right-click Tivoli Enterprise Portal Server and select **Actions**.
3. Select **Advanced Add application support to the TEMS**.
4. On the Add Application Support to the TEMS, select **On a different computer** .

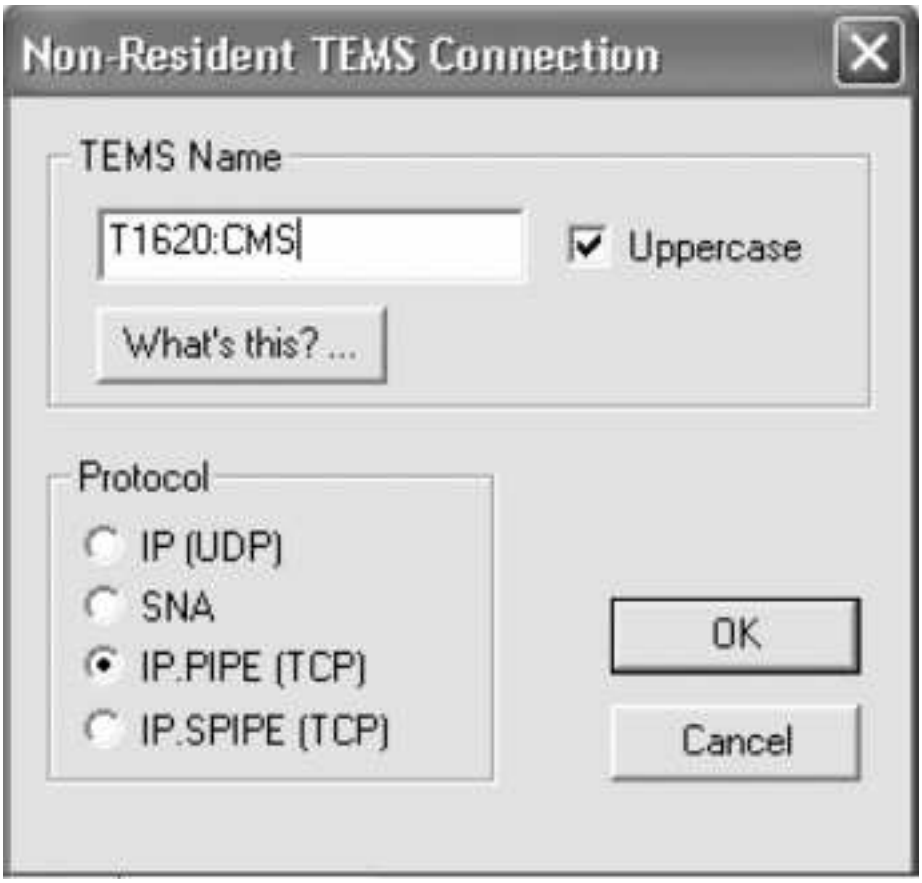


5. The software displays a confirmation window.

Complete one of the following steps:

- If the Tivoli Enterprise Monitoring Server is started, click **OK** and go to the next step.
- If the Tivoli Enterprise Monitoring Server is not started, start it, as explained in "Starting the hub Tivoli Enterprise Monitoring Server on z/OS" on page 30, and click **OK**.

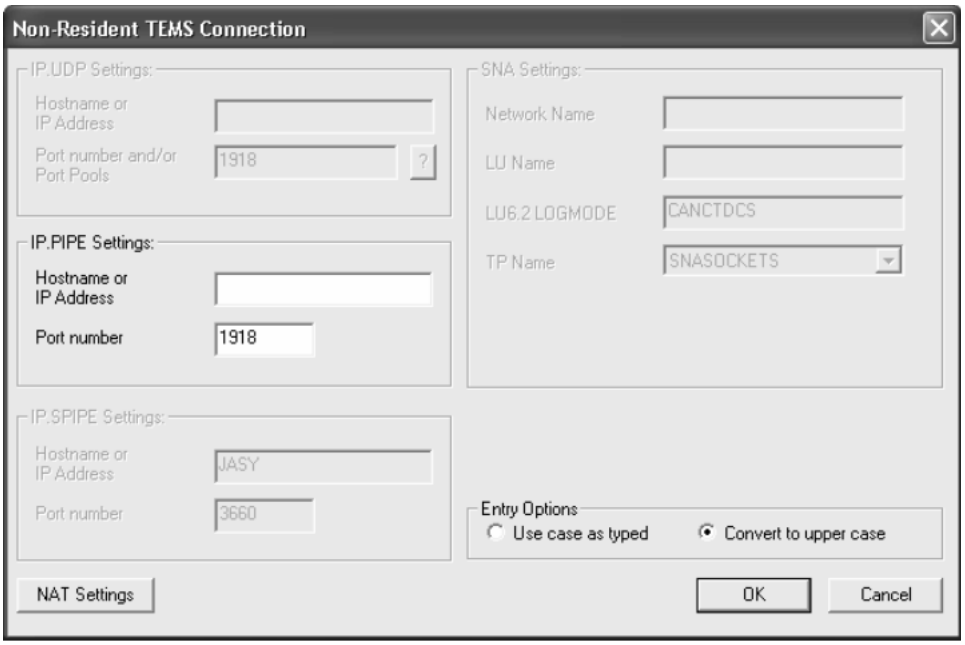
6. Provide the communication protocol and TEMS Node ID for the Non-resident TEMS Connection window and click **OK**.



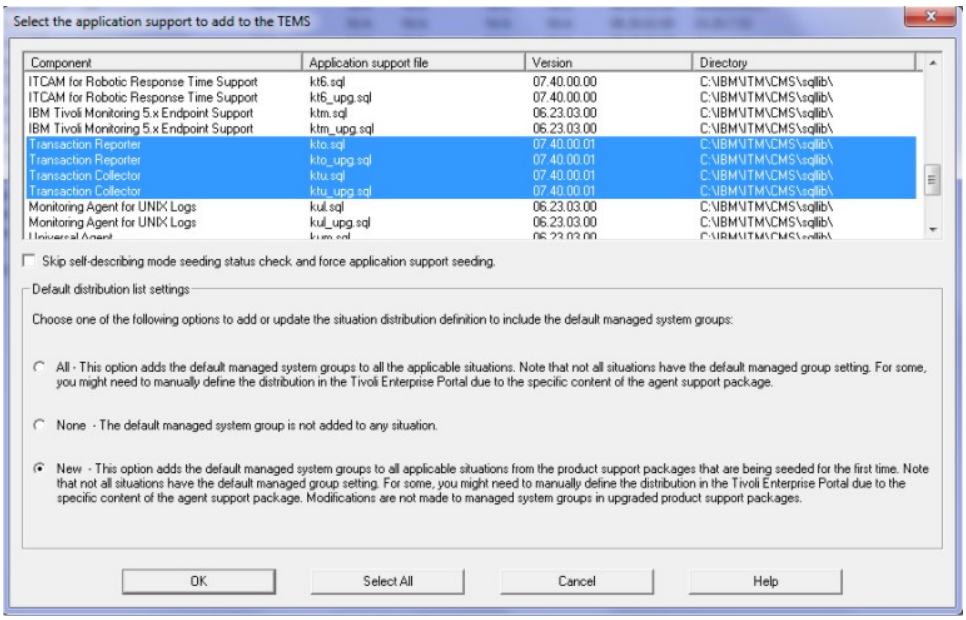
- You can locate the value for the CMS Node ID for the hub Tivoli Enterprise Monitoring Server you configured by examining the `CMS_NODEID` parameter in `KDSENV` member in the `&rhilev.&midlev.RKANPARU` data set on the z/OS system where you configured the Tivoli Enterprise Monitoring Server hub.

- You must select a communication protocol that the Tivoli Enterprise Monitoring Server was configured to support in your environment.

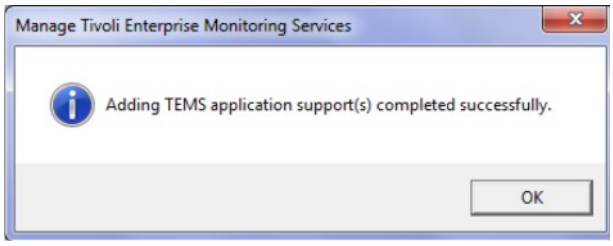
7. Provide the following information for the Non-resident TEMS Connection settings window and click **OK**.
   - **Hostname or IP Address** is the fully qualified hostname or IP address of the Tivoli Enterprise Monitoring Server.
   - **Port #** is the port number that Tivoli Enterprise Portal and Tivoli Enterprise Portal Server uses to communicate with theTivoli Enterprise Monitoring Server on z/OS. The default is 1918.
   - **Entry Options** is the case in which all entries on this screen are processed.



8. On the Select the Application Support to Add to the TEMS, select all of the Transaction Tracking agents and click **OK**. Select all occurrences of the following components to which you want to install support:
   - **Transaction Reporter**
   - **Transaction Collector**



9. Do one of the following:
   - If you see the following confirmation window, adding application support has completed successfully. Go to "Starting the hub Tivoli Enterprise Monitoring Server on z/OS."



   - If you do not see this window, look in the `IBM\ITM\CMS\Logs` file for diagnostic messages to help you determine the cause of the problem.

Starting the hub Tivoli Enterprise Monitoring Server on z/OS

Any time after you have completed the configuration steps for the z/OS components, you can start your hub Tivoli Enterprise Monitoring Server

1. Issue the following command from the z/OS system console (shown using the default procedure name of the started task):

   ```
   S CANSDSST
   ```

   .

2. Verify that the Tivoli Enterprise Monitoring Server has started successfully. You should see the following message displayed on the z/OS system console, or in `SYSLOG`, or in the `RKLVLOG`. The message lists the product and the CMS procedure started task.

   ```
   KO4SRV032 Tivoli Enterprise Monitoring Server (TEMS) startup complete.
   ```

Setting up security for a hub Tivoli Enterprise Monitoring Server running on the z/OS operating system

If you configure the Tivoli Enterprise Monitoring Server on z/OS, use **Security Validation** in the Configuration Tool to enable security for the Tivoli Enterprise Portal logon.

If you set **Security Validation** to **Y**, you must define and authorize the `sysadmin` user ID using RACF or another SAF product. You can define and authorize additional user IDs at this time, but you must define and authorize the `sysadmin` user ID before you can log in to Tivoli Enterprise Portal for the first time.

The Configuration Tool also asks if the Integrated Cryptographic Service Facility (ICSF) is installed. If the IBM Integrated Cryptographic Service Facility (ICSF) is installed and configured on the z/OS system, respond **Y**.

Note: If you specify that ICSF is not installed on this z/OS system (**N**), then the monitoring server uses an alternative, less secure encryption scheme. In addition, communication between a hub monitoring server and the Tivoli Enterprise Portal Server requires ICSF encryption unless you edit the Tivoli Enterprise Management Agent `kfwenv` file.

1. In the **Manage Tivoli Monitoring Service** applet, right-click Tivoli Enterprise Portal Server.
2. Select **Advanced** > **Edit ENV File** from the pop-up menu.
3. Add the following line to the end of the file: `USE_EGG1_FLAG=1`

If ICSF is installed, provide the following additional information to the Configuration Tool:

ICSF load library
: *If ICSF is installed and configured*, specify the ICSF load library that contains the `CSNB*` modules used for password encryption or leave it at the default value of `CSF.SCSFMOD0.`

  If ICSF is not installed on the system, clear this field.

Encryption key
: Specify a unique, 32-byte password for the encryption key or leave it at the default `IBMTivoliMonitoringEncryptionKey`. After this value is written to the key file, you cannot change the encryption key value. The value is case-sensitive. Record the value for the key and store it in a safe place. You must use the same key during the installation of any components that communicate with this monitoring server, such as the Tivoli Enterprise Portal Server or a remote monitoring agent.

If ICSF is not installed, clear the field. For more information about security issues in the Tivoli Management Services environment, see the *IBM Tivoli Monitoring: Installation and Setup Guide*.

Chapter 3. Transaction Tracking for z/OS

Transaction Tracking for z/OS is part of the Transaction Tracking product, providing support for the z/OS operating system.

Transaction Tracking for z/OS consists of the following components:

- Transactions Base
  - Transactions Container - a z/OS started task (STC) that provides basic functionality
  - Transactions Dispatcher - code that runs within the Transactions Container. This code creates Queues to receive Transaction Tracking events, and validates and forwards these events to the Transaction Collector running on a Windows or UNIX Server
- CICS Tracking
- CICS TG Transaction Tracking
- IMS Tracking
- MQ Tracking

The Transactions Dispatcher is located between the caller and the Transaction Collector. It can receive large numbers of events from multiple callers quickly and efficiently using z/OS Cross Memory Services, minimizing elapsed time and overhead. It also minimizes processing within a caller's environment, performing most processing within the Transactions Container. One Transactions Dispatcher must run on every z/OS system where the Transaction Tracking API is used.

The Transaction Collector cannot run on z/OS, and must be located on a Windows or UNIX platform. It receives events from the Transactions Dispatcher over TCP/IP.

Installing and configuring

Before installing Transaction Tracking for z/OS, ensure that you have the required software and understand the installation process.

Software prerequisites

The following software is required:

- z/OS 1.7 or later
- Enterprise COBOL for z/OS 3.4 or later (if using COBOL API)
- Enterprise PL/I for z/OS 3.4 or later (if using PL/I API)
- IBM 31-bit SDK for z/OS Java™ 2 Technology Edition 1.4.2 or later (if using 31-bit Java API)
- IBM 64-Bit SDK for z/OS Java 2 Technology Edition 1.4.2 or later (if using 64-bit Java API)
- High Level Assembler for MVS™ and VM and VSE 1.5 or later (if using HLASM API)

Installing and configuring Transactions Base

Install and configure Transactions Base on your z/OS systems.

Before you begin

Before installing Transactions Base, ensure that you have the required software installed.

Procedure

To install and configure Transactions Base on your z/OS systems:

1. Set up the Transaction Collector.

   Before installing Transaction Tracking on z/OS, set up and verify your Transaction Collector (on a Windows or UNIX system). Ensure that your z/OS system can send TCP/IP data to this computer by using the IP port specified on installation.

2. SMP/E Install Transaction Tracking.

   Perform SMP/E installation as specified in the Transaction Tracking Program Directory. The following libraries are provided for Transactions Base during the SMP/E installation process:

   - SCYTH - C header files for the Transaction Tracking API C language
   - SCYTLOAD - Load modules for Transactions Base,WebSphere MQ, and CICS TG
   - SCYTSAMP - Samples for Transactions Base, WebSphere MQ, and CICS TG
   - SCYTPKGI - SMP/E internal use
   - SCYTUTIN - SMP/E internal use
   - SCYMAUTH - Load modules for IMS Tracking
   - SCYMSAMP - Samples for IMS Tracking
   - TKANMAC - Macros for CICS Tracking
   - TKANMODS - Load modules for CICS Tracking
   - TKANMODR - DFHRPL load modules for CICS Tracking
   - TKANSAM - Samples for CICS Tracking
   - TKANPKGI - SMP/E internal use

3. Determine subsystem Name.

   Every Transactions Container requires a four character z/OS subsystem name. Consult with your Systems Programmer for a suitable subsystem name. Although Transaction Tracking dynamically adds this subsystem on startup, you may want to add it to your z/OS IEFSSNxx parmlib member to document its use. This subsystem name is used later in the installation process.

4. Copy and customize `STC` to `PROCLIB`.

   a. Copy the `SCYTSAMP` member `CYTAPROC` to a PDS in your z/OS `PROCLIB` concatenation, renaming it to the name chosen for the Transactions Container `STC`.

   b. Modify the `JCL` of `CYTAPROC` to point to your Transaction Tracking datasets. Modify the SSNM variable to the subsystem name chosen earlier. If this subsystem is not defined, Transaction Tracking dynamically defines it on startup.

   c. Optional: Modify the CYTAPROC JCL procedure to include a SYSPRINT DD statement.

By default, if messages and trace data are requested, the CYTAPROC sends this data to a dynamically allocated DD that is directed to the JES spool. One DD will be allocated for each started Courier (see "Transactions Dispatcher operation" on page 39 for further information about Couriers). However, if a SYSPRINT DD statement is included in CYTAPROC, all messages and trace data is directed to SYSPRINT. SYSPRINT may then be redirected to a data set. This may be useful if JES spool space is limited and you prefer output to be directed to a data set rather than to spool. Ensure that the SYSPRINT data set is sufficiently large to avoid an x37 ABEND.

For example,

```
//SYSPRINT DD DSN=hlq.SYSPRINT,DCB=(DSORG=PS,LRECL=121), etc
```

d. Ensure that the userid associated with this `STC` has read access to all the Transaction Tracking datasets, and an OMVS segment (allowing it to use UNIX Systems Services).

e. Review and customize the `SCYTSAMP` member `CYTAPARM`. Update the `TTServerString` parameter to point to the IP address and port number of the receiving Transaction Collector. The value must be in the form: `tcp:ip_address:port` – for example `tcp:10.0.0.1:5455`.

f. Review the Transaction Tracking commands to be issued on startup in the `SCYTSAMP` member `CYTACMDS`.

Use the `CYTACMDS` member to change some of the Transactions Container operational defaults. For example, by default the Container sends events to the distributed Transaction Collector, but this default behavior can be changed by specifying `CYTA SEND OFF` in the `CYTACMDS` member. The default behavior is described in Table 5 on page 39.

5. Review `SYS1.PARMLIB` parameters

Unless started with the `REUSASID=YES` parameter, the Transactions Container address space ID (ASID) is not available for reuse, and every restart of the Transactions Container consumes an additional ASID. Review your `SYS1.PARMLIB` IEASYSxx RSVNONR and `RSVSTRT` values. See "Transactions Container operation" on page 38 for more information on ASID reusability for the Transactions Container.

Transactions Base uses a `SCOPE=COMMON` data space (CADS) for each Container STC running on your system. Review `SYS1.PARMLIB` member `IEASYSxx`, parameter `MAXCADS`, to ensure that sufficient resources are available to support the additional CADS allocations.

6. Add `SCYTLOAD` to APF and linklist.

The `SCYTLOAD` library must be added to your z/OS APF list, and your z/OS linklist. If `SCYTLOAD` is not defined to your linklist, all callers of the Transaction Tracking API will require this library in a `STEPLIB`.

7. Optional: Create External Links to Transactions JNI code.

This step is only necessary if you plan to instrument CICS TG Transaction Tracking, zWAS Transaction Tracking, or instrument other applications which call the Java Transaction Tracking API.

Perform the following UNIX Systems Services commands to create external links to the Transactions JNI modules. These links must be located in the libpath of all Java Programs that send events to Transactions:

```
ln -e CYTATJAV libcytapi.so
ln -e CYTATJ64 libcytapi64.so
```

8. Ensure that JAR files are in the Java Classpath. Ensure that all Transactions JAR files are in the Java Classpath of all Java Programs that send events to Transaction Tracking.
9. Ensure that the Transactions Containeraddress space is non-cancellable. Update `SYS1.PARMLIB` member `SCHEDxx` with the following entries:

   PPT PGMNAME(CYTZDRVR)

   NOSWAP

   NOCANCEL
10. Start up the Transactions Container.

    Start the Transactions Container started task.
11. Validate operation.

    Create Transaction Tracking events in z/OS using the Dispatcher command **`CYTA IVP`**, and confirm they are sent to the Transaction Collector.
12. Repeat for all z/OS systems where the API is used.
13. Ensure that the Transactions Container started task is started after every z/OS IPL. Like most monitors, the Transactions Container started task should be started as soon as possible after every IPL, and before any of the Transactions Data Collectors or any Transactions API clients are started.

Transactions Base administration and operation

Transactions BaseTransactions Base consists of two components that together process Transaction Tracking events.

How it works

Transactions Base consists of two components:

- Transactions Container is a z/OS Started task that provides basic functionality for ‘guest' applications, including command processing, message output, and subsystem management.
- Transactions Dispatcher is code running within the Transactions Container that:
  - Creates Queues to receive Transaction Tracking events from the Transaction Tracking API callers.
  - Validates each event received from the Transaction Tracking API callers, and forwards them to the Transaction Collector (residing on another platform).
  - Provides utility functions to allow tracing of events, and management of Queues.

Figure 6 on page 37 shows how the Transactions Base works:

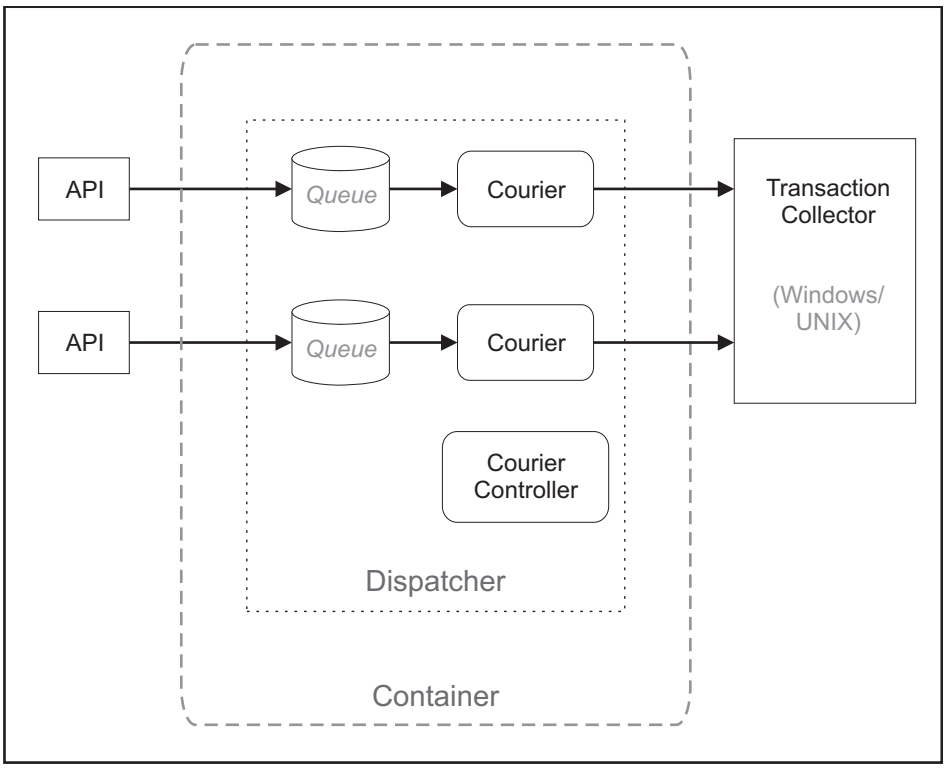


Figure 6. ITCAM for Transactions on z/OS Base

Transaction Tracking events are sent by calling the Transaction Tracking API function `CYTA_track`, specifying the event and a Configuration token obtained from the `CYTA_init` function. It is possible to run more than one Transactions Container on one z/OS image at the same time; however you must use different subsystem names for these containers.

The `CYTA_track` function checks that the Transactions Dispatcher is available, and if so, calls the IBM Tivoli Composite Application Manager for Transactions service. This service validates the structure of the event, and if valid, adds it to a Transactions Dispatcher Queue (using z/OS Cross Memory Services). Control is then returned to the caller. If the structure of the event is invalid, it is not added to a Queue, and control is returned to the caller with the relevant return code. If the Transactions Dispatcher is unavailable, control is returned to the caller with the relevant return code. See "Appendix B" in the *SDK Guide* for a list of return codes.

Each Queue is monitored by one *Courier*, a z/OS UNIX Systems Service (USS) process. Up to 50 Couriers can be running at the same time, each monitoring one Queue. Each z/OS address space is ‘associated' with one Queue after it sends an event. All subsequent events from that address space are sent to the same Queue if it is available. All couriers are controlled by the *Courier Controller* task.

Note: Start one Courier for each API caller. For example, one Courier for each instrumented CICS regions, one Courier for each instrumented WMQ Queue Manager, one Courier for each instrumented IMS region, and so on. This configuration avoids contention between Container queues of instrumented address spaces.

Tip: Start one Container for each ITCAM component. For example, one Container for all CICS regions on an LPAR, and one Container for all WMQ Queue Managers on an LPAR. This configuration provides component isolation and makes debugging easier.

When an event is added to the Queue, the Courier monitoring that Queue is notified. It removes the event from the Queue and forwards it to the Transaction Collector over TCP/IP. If the Transaction Collector is unavailable, all events are queued until contact with the Transaction Collector is made. The Courier attempts to contact the Transaction Collector every 30 seconds until successful, or the Courier is shutdown.

Queues and Couriers are created using the Transactions Dispatcher `CYTA CSTART` command, either via the z/OS console or from the `CYTACMDS` data set at Transactions Dispatcher startup.

Transactions Container operation

The Transactions Container is a z/OS Started task inside which all Transaction Tracking processing is performed.

Start the Transactions Container using the z/OS system console Start command (such as `S CYTAPROC`). Starting the container runs the `CYTZDRVR` program, which accepts an 8–character input parameter. The first four characters specify the code that runs within the Transactions Container, which must be either one of the following two codes:

- `CYTA`: the Transactions Dispatcher code runs inside the Transactions Container.
- `CYTQ`: the Transactions WebSphere MQ monitoring code runs inside the container.

The second four characters specify the name of the subsystem that the Transactions Container uses. The Transactions Container dynamically adds this subsystem on startup if it does not exist already.

To stop the Transactions Container, issue the z/OS system console Stop command (for example, `P CYTAPROC`).

Transactions Container ASID reusability

When the Transactions Container has started, its address space ID (ASID) is unavailable for reuse for the duration of the IPL. The Transactions Container uses z/OS cross memory services in such a way that it must own a cross memory entry table that connects to other address spaces through a system linkage index. z/OS limitations prevent the reuse of ASIDs from any such address space. Hence every time a Transactions Container is restarted, it consumes an additional ASID. Using all available z/OS ASIDs impedes the creation of any new z/OS address space. Review the `PARMLIB IEASYSxx RSVNONR` and `RSVSTRT` parameters and limit the number of times the Transactions Container is restarted.

z/OS 1.9 users with the `PARMLIB DIAGxx` parameter, `REUSASID(YES)` can use the alternate Transactions Container start command `S proc,REUSASID=YES` (for example, `S CYTAPROC,REUSASID=YES`), which allows the Transactions Container to use a z/OS reusable ASID that can be reused after the Transactions Container is shut down.

Operator commands

Commands are issued to the Transactions Container in two ways:

- Using the z/OS system console command `F proc,command` (for example, `F CYTAPROC,DEBUG OFF`)
- By specifying the command in a sequential data set in the `CYTACMDS DD` Container.

The available Transactions Container commands are shown in Table 5.

Table 5. Transactions Container operator commands

Command	Description
`ACT`	Activate the Transactions Container subsystem. The Transactions Container subsystem is activated by default.
`DEBUG ON`	Show debug messages in the Transactions Container `SYSLOG DD`. This option produces a large amount of output. Use this command only if instructed to do so by IBM Software Support personnel.
`DEBUG OFF`	Stop showing debug messages. By default, debug messages are not shown.
`INACT`	Inactivate the Transactions Container subsystem. This stops all Transactions Container and Transactions Dispatcher processing.
`TRACE ON`	Show trace messages in the `SYSLOG DD`. This option produces a large amount of output. Use this command only if instructed to do so by IBM Software Support personnel.
`TRACE OFF`	Stop showing trace messages. By default, trace messages are not shown.
`VER`	Display Transactions Container module version information.

Transactions Dispatcher operation

Providing that `CYTA` is specified as the first four characters of the parameters passed to `CYTZDRVR`, the Transactions Dispatcher code starts up automatically during Transactions Container startup and runs inside the Transactions Container.

Note: No Queues or Couriers are started by default. You must issue one Transactions Dispatcher `CYTA CSTART` command for every Queue or Courier required. You can have up to 50 Couriers running at the same time, however running 3-5 is sufficient for most environments.

Operator commands

Commands are issued to the Transactions Dispatcher in two ways:

- Using the z/OS system console command `F proc, CYTA command`. For example, `F CYTAPROC, CYTA STATUS ALL`.
- By specifying the command in a sequential data set in the Transactions Container `CYTACMD DD`.

The available commands are shown in Table 6 on page 40.

Table 6. Transactions Dispatcher operator commands

Command	Description
`CYTA ADISPLAY asid`	Display the Courier processing events from the address space with Address Space ID (ASID) = *asid*. This command produces output similar to: `CYTA0035I CYTZ PRODIMS (ASID 89) Associated with 83886368` In this example, the Courier with USS process ID 83886368 is processing events from PRODIMS.
`CYTA CACT pid`	Activate the Courier with USS process id = *pid*
`CYTA CACT ALL`	Activate all Couriers.

`CYTA CDISPLAY pid`	Display the status of the Courier with UNIX process id = *pid*. Output similar to the following output is displayed:
	`CYTA0009I CYTZ Courier ID: 83886368 (Active)`
	`CYTA0010I CYTZ Events: 54, ASIDs: 1 Queue: 0`
	
	The output shows the USS Process ID of the Courier, the status (Active), the number of events processed, and the number of z/OS Address spaces currently using the Queue corresponding to the Courier.

`CYTA CDISPLAY ALL`	Display the status of all Couriers.
`CYTA CINACT pid`	Make the Courier with USS process id = *pid* inactive. Use the `CYTA CACT` command to reactivate the process.
`CYTA CINACT ALL`	Make all Couriers inactive.
`CYTA CSTART`	Create a new Queue, and start a new Courier to monitor this Queue.
`CYTA CSTOP pid`	Stop the Courier with process ID = *pid*, and remove the corresponding Queue.
`CYTA CSTOP ALL`	Stop all Couriers, and remove all Queues.
`CYTA DEBUG ON`	Show debug and event information – this option produces a large amount of output. Use this option only if requested to do so by Tivoli support.

`CYTA DEBUG EVENTS`	Show all valid events received by all Couriers. If set, each Courier produces output similar to:
	`05/07/08 00:24:32 CYTAA05I Inbound Event at 1210119872.267234`
	`Horizontal Link (ANY): HLINK`
	`Vertical Link (ANY): VLINK`
	`Vertical Context list:`
	`ApplicationName : IMS1`
	`ComponentName : IMS`
	`TransactionName : TSTAPI`
	`ServerName : SYSPLEX1/MVS1`

`CYTA DEBUG OFF`	Do not show debug information. By default, debug messages are not shown.
`CYTA DISABLE`	Disable the Transactions Dispatcher. This stops the Transactions Dispatcher processing events. Use the `CYTA ENABLE` command to enable the Transactions Dispatcher again.
`CYTA ENABLE`	Enable the Transactions Dispatcher. By default, the Transactions Dispatcher is enabled.

Table 6. Transactions Dispatcher operator commands (continued)

Command	Description
`CYTA IVP`	Send Installation Verification Procedure events to Transaction Collector.
`CYTA SEND ON`	Send all events received to the Transaction Collector. By default, all events are sent to the Transaction Collector.
`CYTA SEND OFF`	Stop sending events to the Transaction Collector. All events received by the Transactions Dispatcher will be discarded. Use for testing only. If, after events have started flowing to the Transaction Collector, the connection to the Transaction Collector fails, the Container will periodically, and continuously, retry the failed connection. Messages CYTAD63I and CYTAD18E will be issued for each failed retry. Entering the `SEND OFF` command at this time stops events flowing to the Transaction Collector but does not stop the connection retry. To stop the connection retry you must either stop and restart the Transactions Container or stop and restart the Couriers. If excessive retry messages are an issue (because of spool space for example), see step 4 in "Installing and configuring Transactions Base" on page 34 for details on how to direct Transactions Container output to disk. If the `SEND OFF` command is coded in the `CYTACMDS` file to be processed at Transactions Container start up, no events flow to the Transaction Collector, and no connection is established or retried.
`CYTA STATUS`	Show status of Courier Controller.
`CYTA VER`	View version information for major Transactions Dispatcher modules.
`CYTA STATUS ALL`	Show status of Dispatcher, Courier Controller, and all Couriers.

Chapter 4. CICS Tracking

CICS Tracking is an extension to the Transaction Tracking for z/OS product, providing CICS support on the z/OS operating system.

The API in CICS tracks:

- ECI traffic that is received from the ITCAM-monitored Java EE servers through CICS Transaction Gateways
- DPL traffic that flows between the CICS systems
- MQSeries traffic that is routed to and from the CICS systems
- SOAP over HTTP traffic that is routed to and from the CICS systems
- CICS Function Shipping and Transaction Routing
- CICS to DB2 traffic
- CICS to IMSDB traffic
- Inbound HTTP traffic serviced by CICS Web Support (CWS)
- Inbound SNA/APPC traffic is represented by pseudo-nodes for the SNA callers

These paths are tracked automatically without any changes to user applications. CICS Tracking uses the Transactions Base infrastructure to send events to the Transaction Collector.

You can also use the Transactions Base API to send your own events from within CICS exits or application programs. CICS Tracking also provides a CICS program that can be called from user programs to send Transactions events. This program automatically populates many of the transactions. See "CICS programming guide and reference" on page 53 for more information.

Installing and configuring CICS Tracking

Before installing and configuring CICS Tracking, ensure that you have the required software and understand the installation process.

Software prerequisite

CICS TS 3.1 or later is required by CICS Tracking.

Installing and configuring CICS Tracking

Install the Transactions Base and then complete the following steps for every CICS region to be monitored to install CICS Tracking:

1. Add the CICS Tracking modules to CICS.
   a. Add the TKANMODR data set to the CICS DFHRPL concatenation.
   b. Add the TKANMOD data set to a RKANMOD DD concatenation within CICS.
2. Add the CYIPINIT program to the CICS Stage 2 PLTPI (after the DFHDELIM statement). This is an example of a CICS PLTPI with CYIPINIT program added:

```
DFHPLT TYPE=INITIAL,SUFFIX=MM
          DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
          DFHPLT TYPE=ENTRY,PROGRAM=CYIPINIT,
          DFHPLT TYPE=FINAL
```

Note: If you want to start CICS Tracking without adding it to the PLT, you can use either of the following methods:

- Run OMEGAMON XE for CICS to start CICS Tracking automatically if OMEGAMON XE for CICS is enabled to use Transaction Tracking
- Run the TTCU CICS transaction with the **START** parameter

3. Define the CICS Tracking programs to CICS. The TKANSAM library member CYISCSDU is a sample batch job that defines these programs.
4. Add the CYISYSIN configuration file to CICS.
   a. Create a PDS parameter dataset with DCB values: **DSORG=PO,RECFM=FB,LRECL=80**. Insert the CYISYSIN parameter entries, keywords, and values into members of this PDS dataset. One member can be used for more than one CICS region.
   b. Each parameter type must be completed on one line, with all parameters separated by a comma. Valid parameters and types are:
      - **TYPE=S** indicates a system configuration card. This must be the first parameter starting in column one.
        - **JOBNAME=*mask*** is used to specify the CICS job name of the CICS region that this parameter applies to.

          Specify the entire job name, or a mask, using an asterisk (*) as a wildcard. For example, CI* applies to all CICS regions beginning with CI. Note that only one asterisk can be specified, and it must be the last character in the mask. If **JOBNAME** is omitted, this parameter applies to all CICS regions.
        - **SUBSYS=*subsys*** specifies the Transactions Base container subsystem name. If omitted, it will default to CYTZ.
        - **DB2TRACK=ON|OFF** specifies whether CICS to DB2 transaction tracking is enabled. The default is **OFF**. See "CICS to DB2 tracking" on page 46 for further information.
        - **IMSTRACK=ON | OFF** specifies whether CICS to IMSDB transaction tracking is enabled. The default is **OFF**. See CICS to IMSDB tracking for further information.
        - **EXITS=YES|NO** specifies if the CICS Tracking exits are to be installed to automatically monitor CICS transactions.

          If **NO**, automatic CICS monitoring does not occur; only events sent through the CICS Tracking wrapper program will be sent. Default is **YES**.
        - **LOCAL=OFF|ON** specifies whether only local CICS transactions are tracked. The default is **OFF**.
        - **FSHPTRACK=OFF|ON** specifies whether tracking of function-shipped or transaction-routed interactions are enabled. The default is **OFF**.

        For example:

        ```
        TYPE=S,JOBNAME=CICS*,SUBSYS=CYT1,EXITS=YES
        ```
      - One of two types of transaction tracking selection criteria: **EXCLUDE** (type-E) criteria and overriding **INCLUDE** (type-I) criteria. Eligible transactions are always tracked unless they match any of the **EXCLUDE** criteria, and the **EXCLUDE** criteria is always compared at task-start time for every transaction. If there is an **EXCLUDE** criteria match, the overriding **INCLUDE** criteria is compared too. If there is also an overriding **INCLUDE** criteria match, the transaction is tracked. However, if no overriding **INCLUDE** criteria match is found, then the **EXCLUDE** criteria match takes effect and the transaction is not tracked.

Transactions Gates and Filters are optional; if no CYISYSIN configuration file is allocated to the CICS region, or if no type-E entries are included in the configuration file, all eligible CICS transactions are tracked. Most 'C'-type CICS system transactions and the CICS Tracking TTCU utility transaction are never eligible for tracking by CICS Tracking regardless of any CYISYSIN configuration file entries.

Do not include CICS background or long-running transactions in ITCAM for Transactions. CICS Tracking automatically excludes all long-running or never-ending C-type system transactions from tracking. Exclude user transactions in CICS that typically run for minutes rather than seconds, or that do not end while CICS is active. Exclude these user transactions using the **TYPE=E** and **TRANSID=** transaction exclude filters.

– **TYPE=E** specifies that this entry is a rule excluding CICS transactions or programs from monitoring:
  - **SYSID=*sysid*** is the CICS SYSID (from the CICS SIT table) that is used to specify the CICS sysid that this rule covers.

    You can specify the entire four-character sysid, or use an asterisk (*) character as a wildcard. For example, CI* applies to all sysids beginning with CI. Note that only one asterisk can be specified, and it must be the last character of the mask. If omitted this rule applies to all CICS regions. The *sysid* must be uppercase.
  - **TRANSID=*txn*** is the parameter that is used to specify the CICS transaction name.

    You can specify the entire four character transaction name, or use an asterisk (*) character as a wildcard. For example CS* would apply to all transactions beginning with CS. Note, that only one asterisk can be specified, and it must be the last character of the mask. The default transaction is CSMI.
  - **PROGRAM=*pgm*** is the parameter that is used to specify the CICS program name.

    You can specify the entire program name, or use an asterisk (*) character as a wildcard. For example AA1* would apply to all programs beginning with AA1. Note, that only one asterisk can be specified, and it must be the last character of the mask. If omitted, specifies all programs. The program name must be uppercase.

  For example:

  ```
  TYPE=E,SYSID=CI01,TRANSID=CS*,PROGRAM=PGM*
  ```

– **TYPE=I** specifies that this is a rule including CICS transactions or programs for monitoring. Parameters for this type are identical to TYPE=E. For example:

  ```
  TYPE=I,SYSID=CI01,TRANSID=CS*,PROGRAM=PROD*
  ```

If a CICS region uses the CYISYSIN configuration file in **Example 1**, the single type-E entry excludes all of the region's tasks from transaction tracking:

Example 1: Transactions Gates and Filters – Exclude All Transactions

```
TYPE=E,SYSID=*,TRANSID=*,PROGRAM=*      Exclude all transactions
```

If a CICS region uses the CYISYSIN configuration file in **Example 2**, the single type-E entry excludes transaction XYZZ only. All other transactions are tracked:

Example 2: Transactions Gates and Filters – Exclude only XYZZ Transactions

```
TYPE=E,TRANSID=XYZZ                     Exclude all XYZZ transactions
```

If a CICS region uses the CYISYSIN configuration file in **Example 3**, the single type-E entry excludes all of the region's tasks from transaction tracking. However, the type-I entries for TRANSIDs TSKA and SYxx override the type-E entry with 'TRANSID=*'. So, ITCAM for CICS tracks TSKA and SYxx tasks in the CICS region but no other transactions.

Example 3: Transactions Gates and Filters – Exclude All Transactions Except for TSKA and SYxx

```
TYPE=E,SYSID=*,TRANSID=*,PROGRAM=*       Exclude all transactions
TYPE=I,TRANSID=TSKA                      Include TSKA transactions
TYPE=I,TRANSID=SY*                       Include SYxx transactions
```

c. Add the CYISYSIN parameter dataset and the member to a CICS region's JCL procedure using the following DD statement format:

```
//CYISYSIN DD DISP=SHR,DSN=your.pds.dataset(member)
```

No other JCL DD statement options, for example `FREE=CLOSE`, should be added to the CYISYSIN DD statement. If the CYISYSIN configuration dataset and member are not created, CICS Tracking uses parameter defaults.

5. Restart your CICS regions to activate CICS Tracking.
   - CICS Tracking has linked to the Transactions Container when the CYIP1043I message is written to the CICS control region syslog.
   - CICS Tracking is operational when the message CYIP1058I is written to the CICS control region syslog.

CICS TG to CICS tracking

If CICS TG Transaction Tracking is used for tracking traffic flowing through a CICS TG daemon instance on z/OS into CICS, ensure that MVS RRS (Resource Recovery Services) is active on the hosting LPAR and that the first CICS region receiving the requests from the CICS TG daemon is initialized with the following SIT table parameter:

```
RRMS=YES
```

CICS to DB2 tracking

When CICS Tracking is installed in a CICS region you can configure CICS and DB2 transaction tracking to be displayed as connections in Transaction Tracking topology workspaces.

To enable CICS and DB2 transaction tracking, using CICS Tracking V7.1.0.2 or later, set the **DB2TRACK=ON** parameter. This parameter reports the accumulated elapsed processing time that is consumed by a CICS transaction to issue DB2 calls.

To display CICS to DB2 transactions in the Transaction Tracking topology workspaces, ensure that the following items are true:

- CICS Tracking is installed in a CICS region.
- The **DB2TRACK=ON** parameter is set. The following options must also be set so that timings are reported:
  - The CICS region's MCT table must be assembled with the **TYPE=INITIAL** option (this cannot be done through CICS RDO):

```
DFHMCT TYPE=INITIAL,           *
  RMI=YES,                     *
  SUFFIX=xx
```

  - The CICS region must be run with the SIT table options, **MN=ON** and **MNPER=ON**, to turn on CICS performance monitoring.

CICS to IMSDB tracking

If CICS to IMSDB tracking is required, set the **IMSTRACK** parameter so that CICS to IMSDB transactions are displayed in the Transaction Tracking topology workspaces.

CICS to IMSDB transaction tracking, which conveys that the CICS and IMSDB transaction and thread nodes can be displayed as a connection in the topology workspaces, occurs when ITCAM for CICS V7.2.0 is installed in a CICS region, and the **IMSTRACK=ON** parameter is set; this is *not* the default setting for this option. The accumulated elapsed processing time that is consumed by a CICS transaction in order to issue the IMSDB calls is reported when the **IMSTRACK=ON** parameter is used.

To display CICS to IMSDB transactions in the topology workspaces, use the following steps:

1. Verify that CICS Tracking is installed in the CICS region.
2. Set the **IMSTRACK=ON** parameter. Set the following options to report the timings:
   a. The CICS region monitor control table (MCT) must be assembled with the **TYPE=INITIAL** option (this cannot be done through CICS RDO):

      ```
      DFHMCT TYPE=INITIAL, *
      RMI=YES, *
      SUFFIX=xx
      ```

   b. The CICS region must be run with the system initialization table (SIT) table options **MN=ON** and **MNPER=ON** to turn on CICS performance monitoring.

Tracking CICS Web Services (SOAP) traffic

To monitor SOAP traffic to or from CICS (CICS Web Services), and if you are using CICS TS 3.1 or CICS TS 3.2, define the CYIPSOAP program as a CICS SOAP Header Handler.

Note: In CICS TS 4.1 or higher, the SOAP traffic is monitored automatically without modification of PIPELINE configuration files.

To track CICS Web Services (SOAP) traffic use these steps:

1. Add the program CYIPSOAP to the CICS Pipeline definition file for all CICS pipelines.

   The following example is a CICS provider pipeline with the ITCAM required changes in bold text:

   ```
   <?xml version='1.0' encoding="UTF-8"?>
   <provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ibm.com/software/htp/cics/
      pipeline provider.xsd ">
      <service>
        <terminal_handler>
           <cics_soap_1.2_handler>
           <headerprogram>
              <program_name>CYIPSOAP</program_name>
            <namespace>http://www.ibm.com/xmlns/prod/tivoli/itcam</namespace
              <localname>*</localname>
              <mandatory>true</mandatory>
              </headerprogram>
           </cics_soap_1.2_handler>
   ```

```
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

The following example is a CICS request pipeline, with the required changes in bold text:

```
<?xml version='1.0' encoding="UTF-8"?>
<requester_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.ibm.com/software/htp/cics/pipeline
   requester.xsd ">
   <service>
     <terminal_handler>
        <cics_soap_1.2_handler>
        <headerprogram>
           <program_name>CYIPSOAP</program_name>
         <namespace>http://www.ibm.com/xmlns/prod/tivoli/itcam</namespace
           <localname>*</localname>
           <mandatory>true</mandatory>
          </headerprogram>
        </cics_soap_1.2_handler>
     </terminal_handler>
    </service>
    <apphandler>DFHPITP</apphandler>
</requester_pipeline>
```

2. Install your pipeline definitions again or restart your CICS regions.

Tracking CICS local transactions

CICS Tracking does not automatically track all transactions in a CICS region. However, in ITCAM for Transactions V7.2.0.1 and later you can enable tracking for additional local transactions if required.

CICS Tracking tracks transactions that use supported facilities such as WebSphere MQ, Web Service, IMS, or DB2.

To track transactions that do not use these facilities, but are started locally in a CICS region, set the **LOCAL=ON** parameter.

After this parameter is enabled, events for the local transaction are sent to Transaction Tracking.

Tracking included CICS function or transaction-routed interactions

CICS Tracking automatically tracks included function requests to tasks for DPL requests only. Similarly, CICS Tracking does not track the relationships of transaction-routed tasks automatically. However, in ITCAM for Transactions V7.2.0.1 and later you can enable further tracking if required.

To track interactions between a Terminal Owning Region (TOR) and Application Owning Region (AOR) for non-DPL traffic, or requests from an AOR to a Resource Owning Region (ROR), set the **FSHPTRACK=ON** parameter.

After this parameter is set, events for non-DPL requests are sent to Transaction Tracking.

Tracking dynamically routed CICS transactions

CICS Tracking automatically tracks and reports on dynamically routed CICS transactions.

The CICS Tracking program CYIPDYP starts the CICS SIT-defined Dtrprogram as CICS Tracking starts up in a CICS region. CICS Tracking then tracks and reports on dynamically-routed CICS transactions.

When CICS Tracking is active in a CICS region, a `CEMT I SYS` command returns with: `Dtrprogram(CYIPDYP)` instead of the `Dtrprogram()` defined in the **`DTRPGM=`** parameter (default `DFHDYP`) in the CICS SIT table.

When invoked, CYIPDYP immediately transfers control to the Dtrprogram that was in place when CICS Tracking was started, where dynamic routing destination decisions are made as usual. CYIPDYP does not interfere with the original Dtrprogram and does not alter any of its transaction routing decisions.

If CICS Tracking is stopped in a CICS region, a `CEMT I SYS` command reports that the original Dtrprogram is the active program and CYIPDYP is no longer the front end for that Dtrprogram.

If CICS Tracking `abends` in a CICS region, CYIPDYP continues to transfer control to the original Dtrprogram and does not interfere with any dynamic routing transaction decisions.

OMEGAMON/CICS and CICS DFHAPPL transaction umbrella names

CICS business applications may run different functions under a single CICS transaction name. These applications are often called 'umbrella' applications. They usually display a general menu of choices or selections, but all their processing is done under the single transaction name which is used to call up the application's main menu. Although efficient in some ways, important details such as metrics and statistics about the application's functions in task history and system performance reports can be difficult to interpret. OMEGAMON/CICS and CICS each have an API feature for replacing the CICS tasks' original transaction names with other, more meaningful transaction names.

A CICS business application can call the `KOCRMCLL` interface to the OMEGAMON/CICS API, `KOCAPI`, or can invoke a CICS EMP (CMF Event Monitoring Point) through the standard EXEC CICS API, to replace a task's original transaction name with another name. When these API calls are placed into different parts of a business application, they assign unique transaction names to the separate parts for more granular task history reporting. Thus, where OMEGAMON/CICS umbrella transaction names or CICS DFHAPPL transaction names are used, CICS Tracking also replaces the original transaction name with the OMEGAMON/CICS umbrella name if there is one. If there is no OMEGAMON/CICS umbrella name but there is a DFHAPPL name, the DFHAPPL name replaces the original CICS transaction name. This aligns CICS Tracking transaction names (nodes) with OMEGAMON/CICS task history reporting where umbrella names are used, and with CICS task history reporting where DFHAPPL names are used.

The specific rules for CICS Tracking transaction name replacements are:

- If an OMEGAMON/CICS umbrella transaction name for a task is used, that name is used in the ITCAM for Transactions Tivoli Enterprise Portal instead of the original CICS transaction name.
- If a DFHAPPL transaction name for a task is used, that name is used in the ITCAM for Transactions Tivoli Enterprise Portal instead of the original CICS transaction name.
- An OMEGAMON/CICS umbrella name always takes precedence over a CICS DFHAPPL name.

Only the KOCRMCLL interface to KOCAPI is supported. User application calls through the `KOCGLCLL` interface to the OMEGAMON/CICS API, `KOCAPI`, are not supported by this feature of CICS Tracking.

For CICS DFHAPPL names, the CICS MCT table must be assembled with a `APPLNAME=YES` parameter to generate the necessary CMF EMPs.

No changes are needed to CICS Tracking to activate original transaction name replacements. However, user applications that use OMEGAMON/CICS umbrella names or CICS DFHAPPL names must be modified to invoke the appropriate product's API. See the OMEGAMON/CICS manuals for instructions and rules about Umbrella Names, and the CICS/TS v3 and later manuals for instructions and rules about DFHAPPL transaction names.

TTCU

TTCU is a 3270-based utility transaction that you can use to change the processing status of CICS Tracking and turn certain options on and off.

These commands may be entered directly into a running CICS region from a CICS screen or automated and provided for all executions of a CICS region by adding them to the CYISYSIN DD data set which was added to the respective CICS regions in an earlier step. The TTCU commands that can be placed into the CYISYSIN file as **`TYPE=S`** parameters are described in step 4 of the procedure in "Installing and configuring CICS Tracking" on page 43.

Table 7 describes the available commands.

Table 7. TTCU commands

Command	Description
`TTCU`	Start the API in the CICS region.
`TTCU,START`	Start the API in the CICS region.
`TTCU,STOP`	Stop the API in the CICS region. This option stops the API in the CICS region by removing it completely. Therefore, a cycle of **`TTCU,STOP`** and **`TTCU`** or **`TTCU,START`** commands can be used to introduce new maintenance for the API without stopping the CICS region too.
`TTCU,TRACE=ON`	Turn on CICS Tracking internal tracing. When this option is on, tracing records are written to the CICS Auxiliary Trace datasets. CICS Auxiliary Trace must also be enabled separately.
`TTCU,TRACE=OFF`	Turn off CICS Tracking internal tracing.

Table 7. TTCU commands (continued)

Command	Description
TTCU,SNAP=ON	Turn on CICS Tracking SNAPs. When requested, an MVS SNAP of the TT event record is taken just before sending the event to the ITCAM for Transactions MVS subsystem. Use this option only if directed to do so by IBM Software Support.
TTCU,SNAP=OFF	Turn off CICS Tracking SNAPs.
TTCU,SASNAME=xxxx	Reset the MVS subsystem name for the Transactions Container subsystem on z/OS.
TTCU,FILTER	Reread the `CYISYSIN` member and reset the processing options.
TTCU,DB2TRACK=ON	Turn on CICS to DB2 tracking.
TTCU,DB2TRACK=OFF	Turn off CICS to DB2 tracking.
TTCU,IMSTRACK=ON	Turn on CICS to IMSDB tracking.
TTCU,IMSTRACK=OFF	Turn off CICS to IMSDB tracking.
TTCU,LOCAL=ON	Turn on CICS local only tracking.
TTCU,LOCAL=OFF	Turn off CICS local only tracking.
TTCU,FSHPTRACK=ON	Turn on tracking for included functions or transaction-routed interactions.
TTCU,FSHPTRACK=OFF	Turn off tracking for included functions or transaction-routed interactions.

Dynamic maintenance procedures

CICS Tracking maintenance is provided through SMPe sysmods.

Using the following procedures, SMPe maintenance can be APPLIED to CICS Tracking target libraries and activated without restarting the CICS region. Inspect all sysmods before APPLYing and activating them for any HOLDACTIONs or other instructions for special handling.

APPLY and activate new SMPe sysmods

When the DFHRPL and RKANMOD DD statements are defined in the CICS region's JCL, this procedure can be used to APPLY and activate new SMPe sysmods without restarting the CICS region:

1. Stop CICS Tracking in the CICS region:
   ```
   TTCU,STOP
   ```
2. SMPe RECEIVE and APPLY sysmods to the CICS Tracking target libraries `*.TKANMOD` and `*.TKANMODR`.
3. Using CEMT, newcopy the programs in the CICS Tracking `*.TKANMODR` library:
   ```
   CEMT SET PROG(CYIPCAPI) NEW
   CEMT SET PROG(CYIPDYP) NEW
   CEMT SET PROG(CYIPINIT) NEW
   CEMT SET PROG(CYIPSOAP) NEW
   ```
4. StartCICS Tracking in the CICS region:
   ```
   TTCU,START
   ```

5. Check the CICS joblog for the message CYIP1058I, indicating that the CICS Tracking startup was successful.

Using CEDA dynamic library definitions

With CICS/TS v4.1 and above, CICS Tracking datasets and JCL DD definitions for DFHRPL and RKANMOD can be replaced with CEDA LIBRARY definitions. Defining and using CEDA dynamic libraries for DFHRPL and RKANMOD DD concatenations in a CICS region is optional. The DFHRPL and RKANMOD DDs can be concatenated with datasets for other products. Before using CEDA dynamic libraries for DFHRPL and RKANMOD, ensure that all products in your DD concatenations can operate with CEDA dynamic library definitions instead of JCL DD statements.

If you choose to use CEDA dynamic library definitions for DFHRPL and RKANMOD, this procedure can be used to APPLY and activate SMPe sysmods without stopping and restarting the CICS region (customize the values for GROUP, RANK and DSNAMExx as necessary for your site).

During the initial deployment of CICS Tracking, define two dynamic libraries using CEDA, TKANMODR and RKANMOD:

```
CEDA DEF LIB(TKANMODR) GROUP(CYIDEFS) RANK(11) DSNAME01(<*.tkanmodr>)
CEDA DEF LIB(RKANMOD) GROUP(CYIDEFS) RANK(12) DSNAME01(<*.tkanmod>)
```

1. Stop CICS Tracking in the CICS region:

   ```
   TTCU,STOP
   ```

2. Using SMP/E, apply RECEIVE and APPLY sysmods to the CICS Tracking target libraries *.TKANMOD and *.TKANMODR.
3. Reinstall the CICS Tracking dynamic libraries in the CICS region: TKANMODR and RKANMOD:

   ```
   CEMT SET LIB(TKANMODR) DISABLED
   CEMT DISCARD LIB(TKANMODR)
   CEDA INSTALL LIB(TKANMODR) GROUP(CYIDEFS)
   CEMT SET LIB(RKANMOD) DISABLED
   CEMT DISCARD LIB(RKANMOD)
   CEDA INSTALL LIB(RKANMOD) GROUP(CYIDEFS)
   ```

4. Using CEMT, newcopy the programs in the CICS Tracking *.TKANMODR library:

   ```
   CEMT SET PROG(CYIPCAPI) NEW
   CEMT SET PROG(CYIPDYP) NEW
   CEMT SET PROG(CYIPINIT) NEW
   CEMT SET PROG(CYIPSOAP) NEW
   ```

5. StartCICS Tracking in the CICS region:

   ```
   TTCU,START
   ```

6. Check the CICS joblog for the message CYIP1058I, indicating that the CICS Tracking startup was successful.

CICS programming guide and reference

Transaction Tracking provides a user-instrumentation API to allow user programs to construct and send events for composite applications not tracked by ITCAM for Transactions data collectors. CICS Tracking expands on this Transaction Tracking API by providing a callable program that partially populates each event from CICS. Use this API to allow user programs to send events that are compatible, and link with, events sent automatically by CICS Tracking. The CYIPAPI and CYIPCAPI functions are used to build ITCAM for Transactions CICS events.

CICS Tracking also provides an operational API that allows CICS applications to initialize and terminate CICS Tracking. The CYIPINIT function is used to start or to stop ITCAM for Transactions in CICS.

Samples provided

The TKANSAM library includes the following samples:

Table 8. TKANSAM library samples

Language	Member	Description
HLASM	CYISAMP1	Sample HLASM program to send events.
HLASM	CYIPSAMP2	Sample HLASM program to send events
HLASM	CYIPSAMP3	Sample HLASM program to send events
COBOL	CYIPSAMP4	Sample COBOL program to send events

Programming reference

All CICS Tracking functions are passed a parameter area. The format of this area is described in Table 9.

Table 9. Formatting for the COMMAREA

Position	Field name	Type	Description
0–7	CYICEYEC	Character	Set this eye catcher to `CYICOMMIC`.
8–11	CYICEBLK	Address	Pointer to the Transaction Tracking event block.
12–15	CYICRTNC	Fullword	Return code.
16–19	CYICRSNC		Reserved.

The TKANMAC library holds macros and copybooks mapping this parameter area:

Table 10. TKANMAC macros and copybooks

Language	Member	Description
COBOL	CYICCMM	COBOL copybook mapping parameter area.
HLASM	CYICOMM	HLASM macro mapping parameter area.

Function: CYIPAPI

Purpose: Send a Transaction Tracking event.

Parameters Required: Pointer to an area mapped by macros CYICCMM and CYICOMM.

Output: Transaction ID, Vertical link ID, Vertical Context and timestamp of the passed event are completed, and event is sent.

Return Codes: As for Transactions Base CYTA_track or CYTA_init functions.

Notes: This function is called via a normal (non-CICS) call:

Example (HLASM):

```
LA    R1,CYICOMMC
ST    R1,PARMLIST
LA    R1,PARMLIST
L     R15,=V(CYIPAPI)
BALR  R14,R15
```

Function: CYIPCAPI

Purpose: Send a Transaction Tracking event.

Parameters Required: Pointer to an area mapped by macros CYICCMM and CYICOMM.

Output: Transaction ID, Vertical link ID, Vertical Context and timestamp of the passed event are completed, and event is sent

Return Codes: As for Transactions Base CYTA_track or CYTA_init functions

Notes: This function is identical to CYIPAPI – however is called via an EXEC CICS LINK command.

Example:

```
EXEC CICS LINK PROGRAM('CYIPCAPI'),         X
   COMMAREA(CYICOMMC),                     X
   LENGTH(=Y(CYICOMML)),                   X
   RESP(RESPONSE)
```

Function: CYIPINIT

Purpose: Initialize or terminate CICS Tracking.

Parameters Required: Pointer to an area with the following format:

Position	Field name	Type	Description
0–7	CYICEYEC	Character	Set this eye catcher to CYICINIT.
8–11	CYICREQ	Address	Function: 1 – initialize 2 - shutdown
12–15	CYICLOC	Address	The address of the entry point of the work area location routine.
16–19	CYICILEN	Fullword	The length of the required task area.
20–24	CYICRET	Fullword	Return code

The TKANSAM member CYICINIT is a HLASM macro mapping this area.

Output: Messages indicating success or otherwise sent to CICS CSSL.

Return Codes: 0 – Function successful. Otherwise - an error occurred. See CICS Tracking messages: CYIP* in the *IBM Tivoli Composite Application Manager for Transactions Troubleshooting Guide* for more information.

Notes:

- This operational API provides an alternative to initializing CICS Tracking by adding an entry to the PLTPI. It is not required for normal operation of CICS Tracking.
- Callers must pass the address of a routine that will return the address of a work area obtained by the user. This routine is branched to by CICS Tracking. The required length of this work area is returned in the CYICILEN field by CYIPINIT; the caller must obtain this work area after the CYICILEN call.

Example (HLASM):

```
EXEC CICS LINK PROGRAM('CYIPINIT'),        X
     COMMAREA(CYICINIT),                   X
     LENGTH(=Y(CYICINIT)),                 X
     RESP(RESPONSE)
```

Chapter 5. CICS TG Transaction Tracking

CICS TG Transaction Tracking monitors transactions as they flow through CICS Transaction Gateway (CICS TG) components and generates tracking data about those transactions.

CICS TG Transaction Tracking consists of a data collector which runs within the process or address space of the CICS TG component and captures transaction flow information about Gateway daemons and client applications. The flow information is then forwarded to the Transaction Collector for integration with events generated by data collectors from other domains, for example CICS. This data collector is available on both z/OS and distributed platforms. A data collector must be installed on every system that runs the CICS TG components that you want to track.

CICS TG Transaction Tracking is available in ITCAM for Transactions V7.1.0.1 and later. Support for IPIC-based transactions is available in ITCAM for Transactions V7.3.0.1 and later.

CICS TG Transaction Tracking is supported on the following components:

- CICS TG V7.1 and later on both z/OS and distributed systems
- WebSphere Application Server V6.1 and later on both z/OS and distributed systems with CICS TG-supplied Resource Adapters V7.1 and later installed
- Stand-alone Java client applications compiled against CICS TG application classes V7.1 and later

Note: Not all transaction types and protocols are supported. The following table shows the domains supported by CICS TG Transaction Tracking and the transaction types and protocols that can be correlated across the domains it interacts with.

Table 11. CICS TG Transaction Tracking - domain and transactions supported

Domain	**Transaction tracking through CICS supported**
Remote Mode (3-tier) For example, WebSphere Application Server or stand-alone application to CICS TG Gateway Daemon to CICS.	ECI SyncOnReturn (EXCI and IPIC protocols) Extended Mode ECI (IPIC protocol only) XA/2-phase commit (EXCI and IPIC protocols)
Local Mode (2-tier) For example, WebSphere Application Server or stand-alone application to CICS.	ECI SyncOnReturn (IPIC protocol only) Extended Mode ECI (IPIC protocol only) XA/2-phase commit (IPIC protocol only)

Flows are not correlated between the Gateway daemon (remote mode) or client applications (local mode) and CICS for all transaction types if the protocol used to connect to the target CICS server is ECI over TCP/IP or SNA, or if the transaction type is Extended Mode ECI and the protocol used is EXCI. Flows are not correlated between the Gateway daemon and CICS if a user replaceable module (DFHXCURM) is used that routes EXCI requests to a CICS region other than the region to which the original request would have flowed.

Where flows cannot be correlated, either because the interaction between the domains is not supported (for example, a flow between CICS TG and CICS over SNA protocol) or one part of the interaction has not been instrumented (for example, the CICS TG Transaction Tracking data collector is enabled but the CICS data collector is disabled), details of the connection can be displayed using pseudo nodes. A pseudo node represents the untracked part of the transaction and appears on a topology view as a standard icon connected to other nodes with a dashed line instead of a solid line.

For CICS TG Transaction Tracking, pseudo nodes can be used to identify flows to or from clients that cannot be instrumented (for example non-Java ECI V2 clients) and flows between CICS TG components and TXSeries server instances.

Pseudo interactions are enabled by default. If you do not want to display pseudo interactions, you can disable them. See Transaction Reporter agent configuration parameters for further information.

In the workspaces within the Tivoli Enterprise Portal, events generated by client applications, for example a J2C connection factory within WebSphere Application Server or a stand-alone application, appear as component type `CTG Client`. Events generated by a Gateway daemon appear as component type `CTG Gateway`. In a remote mode configuration, you will see a CTG Client instance link with a CTG Gateway instance, which in turn stitches to a CICS instance. In a local mode configuration, you will see a CTG Client instance stitch directly with the CICS instance and there will be no CTG Gateway instance.

If you are monitoring a J2C connection factory on a WebSphere Application Server with ITCAM for Application Diagnostics configured, the events generated by the CICS TG Transaction Tracking data collector can stitch with ITCAM for Application Diagnostics events. If configured, you will see the events generated by ITCAM for Application Diagnostics stitched to a CTG Client instance.

The CICS TG Transaction Tracking data collector writes log messages to the standard error log stream of the CICS TG component it is tracking. You can also configure the data collector to write log messages to a file.

The remainder of this section deals with the installation and configuration of CICS TG Transaction Tracking on z/OS systems. See Preparing CICS TG Transaction Tracking in the Installation and Configuration Guide for further information.

To use CICS TG Transaction Tracking:

- Install CICS TG Transaction Tracking on z/OS systems

  Install CICS TG Transaction Tracking monitoring exits on all z/OS systems running CICS TG components with transactions that you want to track.
- Configure the CICS TG Transaction Tracking data collector

  Configure the behavior of CICS TG Transaction Tracking by setting the parameters in the default configuration file. You can also specify a different configuration file if required.
- Enable CICS TG Transaction Tracking in a Gateway daemon

  To monitor CICS TG transactions in a Gateway daemon, customize the Gateway daemon configuration to enable CICS TG Transaction Tracking.
- Enable CICS TG Transaction Tracking in WebSphere Application Server

To monitor CICS TG transactions from a WebSphere Application Server instance, customize the Java Platform Enterprise Edition Connection Architecture (J2C) connections to enable the CICS TG Transaction Tracking data collector.

- Enable CICS TG Transaction Tracking for stand-alone applications

  Applications that use CICS TG JavaGateway client class can also be configured to use CICS TG Transaction Tracking.

Installing CICS TG Transaction Tracking on z/OS systems

Install the CICS TG Transaction Tracking on all z/OS systems running either CICS TG client applications or Gateway daemons with transactions that you want to track. The CICS TG Transaction Tracking data collector is provided as part of the Transactions Base FMID.

Before you begin

You must have permission to modify HFS/zFS file systems.

Procedure

To install CICS TG Transaction Tracking, complete the following steps:

1. Install the Transactions Base. If you have not already done so, install the Transactions Base component. Include the steps to create external links to the Transaction JNI code, which creates create several files within an HFS or zFS directory. This directory is referred to as *ctgt.install.dir* in this guide. You will need the location of this directory for later configuration of the CICS TG components. If you are unsure of the location of *ctgt.install.dir*, search for the `cytgtt.jar` file. See "Installing and configuring Transactions Base" on page 34 for further information.
2. Create external links to the CICS TG Transaction Tracking native modules.

   In addition to the external links created during the installation of the Transactions Base, run the following UNIX Systems Services commands in the same HFS or ZFS directory where the Transactions Base is installed (*ctgt.install.dir*) to create external links to the native modules:

   ```
   ln –e CYTAGJNI libcytattdcjni.so
   ln –e CYTAGJN6 libcytattdcjni_64.so
   ```
3. Create the default CICS TG Transaction Tracking configuration file. Copy the SCYTSAMP member CYTAGCFG to the HFS or zFS directory which has read and write permissions, and rename this file to `cytgexitconfig.cfg`. If this is not the same directory as used in the previous steps, where the CICS TG Transaction Tracking JAR files and native files are installed, you will need to know the location of this directory for later configuration of the CICS TG components. This directory is referred to as *ctgt.config.dir* in this guide.

What to do next

After installation on all required systems, next configure all data collectors so that CICS TG Transaction Tracking can interact with the rest of your transaction tracking environment.

Configuring the CICS TG Transaction Tracking data collector

You can customize the behavior of CICS TG Transaction Tracking by setting the properties in the CICS TG Transaction Tracking configuration file.

Tip: Review the default settings and make any required changes before enabling CICS TG Transaction Tracking.

See the following table for a description of the properties that can be set in the CICS TG Transaction Tracking configuration file. The configuration file is called `cytgexitconfig.cfg` on z/OS and `ttdcctgexits.cfg` on distributed systems.

Tip: If the Transaction Collector you want to send event data to is not located on the same system as the CICS TG Transaction Tracking data collector you will need to at least update the setting for Transaction Collector location (**`TTServerAddr`**).

Table 12. CICS TG Transaction Tracking properties

Parameter	Default value	Description
`TTServerAddr`	Distributed: `tcp:127.0.0.1:5455` z/OS: `ssn:CYTZ`	The address of the Transaction Collector to which events are sent. For IPv6, the address on distributed systems is `tcp:[::1]:5455`.
`EnableFileLogging`	`false`	If `true`, log to a file on the file system. By default, this is a file in the same directory as the `ctg-tt.jar` file on distributed systems or `ctgtt.jar` file on z/OS systems.
`LogFile`	Windows systems: `c:\temp\ctg-tt.log` All other platforms: `/var/log/ctg-tt.log`	Specifies the file to which messages are logged, if **`EnableFileLogging`** is `true`, and **`LogFile`** is set. **Note:** Ensure that the log file already exists before you set this parameter. Windows systems require a double backslash in path names.
`LogLevel`	`info`	Controls the amount of logging information generated by the exits. **`LogLevel`** can be one of the following values: • `debug` • `info` • `warning` **Note:** Debug messages are logged only to file. **`EnableFileLogging`** must be `true` for you to see debug messages.
`EnableStderrLogging`	`true`	If `true`, log to standard error messages.
`AlwaysLogExceptions`	`false`	When `false`, the monitoring exits attempt to limit the number of exceptions logged within a period of time, so some exceptions may not be logged. If the parameter is `true`, exceptions are always logged.

Table 12. CICS TG Transaction Tracking properties (continued)

Parameter	Default value	Description
UseReverseDNS	false	Affects how the monitoring exits determine the server name for inclusion in TT API events. If **UseReverseDNS** is true, the exits first attempt to do a reverse lookup of the address of the interface on which the flow was received or sent. If **UseReverseDNS** is false and **UseLocalHostName** is true, the exits use the local host name. If this cannot be determined, or **UseLocalHostName** is also false, the exits use the textual representation of the IP address of the interface on which the flow request was received or sent.
UseLocalHostName	true	
LinkWithClientLocation	false	Controls whether the client IP address should be included in horizontal links between CICS TG clients and gateways, and must be set identically on all clients and gateways. Do not alter this parameter unless you understand the implications.
InternalQueueSize	10000	Defines the internal queue size for holding unprocessed CICS TG request exit data. Increase this value if you experience error messages indicating that the internal queue is full. Do not alter this parameter unless you understand the implications.

Enabling CICS TG Transaction Tracking in a Gateway daemon

To monitor CICS TG Transaction Tracking from a Gateway, customize the configuration settings of each Gateway daemon to use the CICS TG Transaction Tracking data collector.

Procedure

To enable CICS TG Transaction Tracking in a Gateway daemon, complete the following steps:

1. If you have not already added SCYTLOAD to your linklist as part of the Transactions Base configuration, either do so or modify the STEPLIB in the CICS TG JCL to include the SCYTLOAD library.
2. Modify the CICS TG environment file (typically referred to as ctgenv):
   a. Add the Transactions Base and CICS TG Transaction Tracking JAR files to the CLASSPATH statement. For example:

```
CLASSPATH=
ctgt.install.dir/cytgtt.jar:\
ctgt.install.dir/cytagjar.jar:\
ctgt.install.dir/cytapi4j.jar
```

If the CLASSPATH statement does not appear within the CICS TG environment file, create it.

b. Add the directory holding the external links defined within the installation to the LIBPATH statement. For example:

```
LIBPATH=ctgt.install.dir
```

If the LIBPATH statement does not appear within the CICS TG environment file, create it.

c. If your CICS TG data collector configuration file cytgexitconfig.cfg, is located in a different directory to the CICS TG Transaction Tracking JAR files, that is, ctgt.config.dir is a different directory to ctgt.install.dir, add the path to the CICS TG data collector configuration file as a JVM startup option. Add the following option:

```
CTGSTART_OPS=-j-Dcom.ibm.transactions.ctg.config= ctgt.config.dir/
cytgexitconfig.cfg
```

If the CTGSTART_OPS statement does not appear within the CICS TG environment file, create it.

3. Modify the CICS TG configuration file (typically referred to as ctg.ini). Add the following line to enable CICS TG Transaction Tracking data collector:

```
requestexits=com.ibm.itcam.transactions.domains.ctg.TTDCCtgMonitoringExit
```

If you already use the **requestexits** parameter, add the CICS TG Transaction Tracking data collector after your existing **requestexits** parameter value, separating the exits by a comma.

4. If they are not already set, set the **applid** and **applidqualifier** parameters in the **PRODUCT** section of the CICS TG configuration file.

When appropriately set, data from individual Gateway daemons is aggregated separately by the transactions framework. If they are not set, all transaction data is aggregated into the generic CTG.ANON identifier.

For IPIC-based transactions to correlate between the CICS TG and CICS, a value must be set for **applid**.

5. Restart the Gateway daemon to activate the CICS TG Transaction Tracking data collector.

Results

CICS TG Transaction Tracking is operational when the message CYTG009I is written to the Gateway daemon syslog.

Enabling CICS TG Transaction Tracking in WebSphere Application Server

To monitor transaction flows from a WebSphere Application Server instance, customize the Java Platform Enterprise Edition Connection Architecture (J2C) connections to use CICS TG Transaction Tracking data collector.

Before you begin

Before you can customize WebSphere Application Server to use CICS TG Transaction Tracking, the following resources are required:

- One or both of the CICS TG 7.1 or higher resource adaptors, cicseci.rar and cicseciXA.rar installed in WebSphere Application Server.

- One or more J2C connection factories that have been created to use the listed resource adaptors. These connection factories must be individually configured to use the CICS TG Transaction Tracking data collector.

Procedure

To set the J2C connection factories to use CICS TG Transaction Tracking:

1. If you have not already added SCYTLOAD to your linklist as part of the Transactions Base configuration, either do so or modify the STEPLIB in the WebSphere Application Server JCL to include the SCYTLOAD library.
2. Add the CICS TG Transaction Tracking JAR files and native files to Resource Adaptor properties in WebSphere Application Server.
   a. In the WebSphere Application Server administration console, use the navigation pane and select **Resources** > **Resource Adaptors** > **Resource Adaptors**.
   b. Select a CICS TG resource adaptor.
   c. Add the following text to the **Class path** field:

      ctgt.install.dir/cytgtt.jar
      ctgt.install.dir/cytagjar.jar
      ctgt.install.dir/cytapi4j.jar
   d. Add the folder containing the external links defined within the installation (`ctgt.install.dir`) to the **Native path** field.
   e. Repeat this step for the other resource adaptor if both are installed.
3. Set each monitored J2C connection factory to use CICS TG Transaction Tracking.
   a. In the WebSphere Application Server administration console, use the navigation pane and select **Resources** > **Resource Adaptors** > **J2C Connection Factories**.
   b. Select a J2C connection factory that you want to monitor.
   c. Select **Custom Properties** for that connection factory.
   d. Set the **RequestExits** property to the following value:

      `com.ibm.itcam.transactions.domains.ctg.TTDCCtgMonitoringExit`
   e. Set the **Applid** and **ApplidQualifier** values if they are not already set.

      When appropriately set, data from individual J2C connection factories is aggregated separately by the transactions framework. If they are not set, all transaction data is aggregated into the generic `CTG.ANON` identifier.

      For IPIC-based transactions to correlate between the CICS TG and CICS, a value must be set for **`applid`**.
   f. Repeat this step for all the CICS TG J2C connection factories that you want to monitor.
4. If your CICS TG data collector configuration file `cytgexitconfig.cfg`, is located in a different directory to the CICS TG Transaction Tracking JAR files, that is, `ctgt.config.dir` is a different directory to `ctgt.install.dir`, add the path to the CICS TG data collector configuration file as a JVM startup option. In the WebSphere Application Server administration console, complete the following steps:
   a. In the navigation pane select **Servers** > **Server Types** > **WebSphere application servers** > *server_name*.
   b. In the **Server Infrastructure** section, select **Java and process management** > **Process definition** > **Java virtual machine** > **Custom properties**.

c. Set a new **Custom Java Virtual Machine** property with the following values:

```
Name com.ibm.transactions.ctg.config
Value ctgt.config.dir/cytgexitconfig.cfg
```

5. Save the changes and restart WebSphere Application Server.

Results

The CICS TG Transaction Tracking data collector is initialized when the first work request is made to a J2C connection factory configured with the CICS TG Transaction Tracking data collector. Confirm that CICS TG Transaction Tracking is initialized by checking for message CYTG009I in the WebSphere Application Server log.

What to do next

If you are monitoring a J2C connection factory on a WebSphere Application Server instance with ITCAM for Application Diagnostics configured, the Transaction Tracking API events generated by CICS TG Transaction Tracking monitoring exits stitch with the ITCAM for Application Diagnostics events.

To enable stitching between WebSphere Application Server and CICS TG generated Transaction Tracking events using tokenless CICS TG Transaction Tracking, some configuration is required to your ITCAM for Application Diagnostics configuration:

1. Add properties to the DC/runtime/{platform.node.server}/custom/toolkit_custom.properties file to:
   - Enable integration with Transaction Tracking:

     ```
     com.ibm.tivoli.itcam.dc.ttapi.enable=true
     com.ibm.tivoli.itcam.dc.ttapi.ttas.transport=tcp:IP address:port number
     ```

   - Enable CICS TG instrumentation:

     ```
     com.ibm.tivoli.itcam.dc.ctg.enablectg=true
     ```

   - Enable integration with tokenless CICS TG Transaction Tracking

     ```
     com.ibm.tivoli.itcam.dc.ttapi.ctg.tokenless.enabled=true
     ```

2. Customize the DC/runtime/{platform.node.server}/custom/ctg.filters file to define which CICS TG requests are to be tracked without embedded tokens using tokenless CICS TG Transaction Tracking.

See Integrating the Data Collector with ITCAM for Transactions for further information.

Enabling CICS TG Transaction Tracking in stand-alone applications

Applications that use CICS TG JavaGateway client class can also generate Transaction Tracking events. Enable the CICS TG Transaction Tracking data collector within the client application by completing these steps.

Procedure

To enable CICS TG Transaction Tracking in stand-alone applications:

1. Modify the CLASSPATH and PATH environment variables for either the system or the user ID that runs the client application process:
   a. Add the CICS TG Transaction Tracking JAR files to the CLASSPATH environment variable. For example:

```
CLASSPATH=
ctgt.install.dir/cytgtt.jar:\
ctgt.install.dir/cytagjar.jar:\
ctgt.install.dir/cytapi4j.jar
```

b. Add the folder containing the external links defined within the installation (*ctgt.install.dir*) to the PATH environment variable.

2. Enable the CICS TG Transaction Tracking data collector using one of the following methods:
   - Set the following JVM property when you start the client application:

     ```
     -DrequestExits=com.ibm.itcam.transactions.domains.ctg.TTDCCtgMonitoringExit
     ```

   - Enter the following code and recompile the client application:

     ```
     JavaGateway gateway = new JavaGateway();
     ...
     gateway.setRequestExits
     ("com.ibm.itcam.transactions.domains.ctg.TTDCCtgMonitoringExit");
     ```

3. Restart the client application.

Results

The CICS TG Transaction Tracking data collector is initialized when a JavaGateway object is created by the client application. Confirm this by checking for message CYTG009I in the client application log.

Chapter 6. IMS and IMS Connect Tracking

IMS and IMS Connect Tracking is an extension of the Transaction Tracking for z/OS product, providing support for IMS on the z/OS operating system.

IMS Tracking tracks the following transactions:

- IMS transaction traffic received from ITCAM-monitored Java EE servers through IMS Connect/OTMA.

  Requires ITCAM for Application Diagnostics (was ITCAM for WebSphere) V6.1 with at least Fix Pack 4 installed to trace Java EE and IMS Connect traffic.
- IMS Connect transaction traffic into IMS.
- VTAM®, BTAM, and APPC (non-OTMA transactions) transaction traffic executing in IMS environments.
- MQSeries traffic routed to and from IMS systems, if there is no MQ Broker between the participating servers.

  Requires MQ Tracking for z/OS.
- Shared-Queues and MCS Link traffic flowing between IMS single or sysplex environments.
- DB2 Database SQL traffic from IMS applications. IMS Tracking does not support dynamically added DB2 subsystems.

These paths are tracked automatically without any changes to user applications.

Note: IMS Tracking does not trace IMS Fast Path (IFP) applications or IMS DBCTL (CICS transaction threads).

IMS Tracking uses the Transactions Base infrastructure to send events to the Transaction Collector.

You can also use the Transactions Base API to send your own events from within IMS exits or application programs. See "Transactions Base administration and operation" on page 36 for more information.

Installing and configuring IMS Tracking

Before installing and configuring IMS Tracking, ensure that you have the required software and understand the installation process. The IMS Connect Data Collector is installed with IMS Tracking. It requires additional configuration of the IMS Connect startup procedures. IMS Connect is included as part of each IMS release.

Software prerequisites

The following software is required:

- IMS 8.1, 9.1, 10, or 11 and IMS Connect 9.1, 10, or 11. IMS V12 is supported by program temporary fix (PTF):
  - For V7.3, UA62734
  - For V7.2, UA62733
- Transactions Base V7.4

Note: IMS Tracking V7.4 does not coexist with ITCAM for IMS V6.1 with Managing Server (MS). You must use IMS Tracking V7.1 or later to send transaction monitoring data to ITCAM V6.1 Managing Server/Virtual Engine GUI.

Installing and configuring IMS Tracking

To install and configure IMS Tracking:

1. Install the Transactions Base.

   Install the Transactions Base product.
2. Install IMS Tracking V7.4.
3. Modify and run CYMLINK.

   Run the SCYMSAMP sample job CYMLINK. This job creates a copy of the SCYMAUTH loadlib in a new name for each IMS version and re-links IMS exits DFSMSCE0 and DFSYIOE0 for the appropriate version. If your site runs multiple IMS versions, you must bind these exits into separate libraries for each version.
4. Complete the IMS Exit coexistence steps.

   IMS Tracking uses some standard IMS exits. If your site also uses these exits, perform the additional installation steps described in "Coexisting with IMS exits" on page 69. These IMS Exits must be linked into the IMS version specific SCYMAUTH load libraries created in step 3.
5. Add `IMS`*`xxx`*`.SCYMAUTH` to the APF list, where *xxx* is the IMS version number (910, A10, or B10).

   The `IMS`*`xxx`*`.SCYMAUTH` library must be added to your z/OS APF list.
6. Implement IMS Tracking Command Interface:
   a. Add `IMS`*`xxx`*`.SCYMAUTH` to the Transactions Container STEPLIB.
   b. Change the Transactions Container JCL **`PARMS`** from `CYTA` to `CYMI`. For example:

      ```
      CYTAPROC PROC SSNM='CYTZ',PARMS='CYMI'
      ```
   c. Restart the Transactions Container.
7. Add `IMS`*`xxx`*`.SCYMAUTH` to `IMS STEPLIB`, where *xxx* is the IMS version number (910, A10, or B10).

   Add the `IMS`*`xxx`*`.SCYMAUTH` library to the `STEPLIB DD` concatenation of every IMS control region to be monitored. An example procedure is provided in the `SCYMSAMP` member `CYM$PROC`.

   Note: This data set must be installed before the IMS SDFSRESL library. The ITCAM for IMS V6.1 SCYNAUT3 library cannot be used in the IMS startup procedure (on the STEPLIB DD). IMS Tracking V7.4 does not support ITCAM for WebSphere v6.1 Managing Server (MS) and Virtual Engine (VE) GUI. Only V7.4 SCYMAUTH is required.
8. Add CYMIMSIN DD.

   The DD must point to a valid file containing IMS Tracking startup parameters. Only one version of `CYM$PATH` is required unless you want to specify different startup parameters for each IMS region. `CYM$PATH` is not IMS-version dependent. See the `SCYMSAMP CYM$PATH` member for more information about these input parameters.
9. Customize CYM$PATH.

   If you customize IMS Tracking startup parameters, review and modify the `SCYMSAMP CYM$PATH` member. See this member for valid parameters, defaults, and instructions on customizing this file.

Note: By default, DB2 monitoring is turned off (**DB2=OFF**). You must change this parameter to **DB2=ON** to enable DB2 monitoring. If you start IMS with **DB2=OFF** you cannot turn it on using the modify command (`F CYTAPROC, CYMI imsid DB2 ON`). Start IMS with **DB2=ON** to enable DB2 monitoring. You can then turn DB2 off and on using the modify command.

10. Restart your IMS regions.

    Restart your IMS regions to activate IMS Tracking. IMS Tracking is operational when the message CYM1012I is written to the IMS control region syslog.

    Note: Ensure that the Transactions Container is active before each IMS is started.

Coexisting with IMS exits

IMS Tracking uses the following IMS exits to automatically monitor IMS transactions:

- IMS TM and MSC Message Routing and Control user exit routine (DFSMSCE0).
- IMS OTMA Input/Output Edit exit routine (DFSYIOE0).
- IMS Connect User Exit (HWSTECL0)

The IMS DFSMSCE0 exit point supersedes the following older IMS exit points:

- Input Message Routing (DFSNPRT0)
- Link Receive (DFSCMLR0/DFSCMLR1)
- Program Routing (DFSCMPR0)
- Terminal Routing (DFSCMTR0)

Tip: These exits work only with IMS V8 or V9. Convert these exits to use the TM and MSC Message Routing and Control User exit routine (DFSMSCE0) before running them with IMS V10 or higher.

If you use any of these exits at your site, run the SCYMSAMP job described in Table 13.

Table 13. SCYMSAMP jobs to run with specific exits

If you use this exit:	Run this SCYMSAMP job:
DFSYIOE0	CYM$IOE0
DFSCMLR0/DFSCMLR1	CYM$MLR0 and/or CYN$MLR1
DFSCMPR0	CYM$MPR0
DFSCMTR0	CYM$MTR0
DFSNPRT0	CYM$PRT0
DFSMSCE0	CYM$SCE0
HWSTECL0	CYM$ECL0

Notes:

- Use these jobs to copy your DFS* modules to the IMS Tracking load libraries, renaming them to ZFS*. The IMS Tracking module automatically calls this module.
- Modify these jobs to put the ZFS* modules into the IMS release specific SCYMAUTH load library. For example, change `SYSLMOD DSN=`**TDIMS.HCYM73I.SCYMAUTH** to **TDIMS.HCYM73I.IMS*xxx*.SCYMAUTH**, where *xxx* is the IMS release (910, A10 or B10).

- Use these jobs to link ZFS* modules into IMS version-specific SCYMAUTH load libraries created by CYMLINK.
- Run these jobs every time you modify your IMS exits.
- Restore your original environment by removing the IMS Tracking libraries from the IMS control region's STEPLIB DD and comment out or remove the CYMIMSIN DD statement.
- See step 5 in "Installing and configuring IMS Connect Tracking" for more information on IMS Connect user exits.

Coexisting with IMS Tools Generic TM and MSC Message Routing Exit (GEX)

Before installing and configuring IMS Tracking, ensure that you have the required software and understand the installation process. The IMS Connect Data Collector is installed with IMS Tracking. It requires additional configuration of the IMS Connect startup procedures. IMS Connect is included as part of each IMS release.

The IMS Tools Generic TM and MSC Message Routing Exit (also referred to as Generic MSC Exit with product prefix GEX) enables multiple copies of the IMS MSC Exit routine (DFSMSCE0) to exist and to be driven within a single IMS environment. If you are using the IMS Tools Generic MSC Exit (GEX) driver to allow multiple instances of the IMS MSC DFSMSCE0 exit to be loaded and called in one IMS Control Region, do not link your version of IMS DFSMSCE0 exit to ZFSMSCE0. Instead, add your DFSMSCE0 exit and ITCAM for Transaction Tracking IMS's DFSMSCE0 exit CYMMSCyz for the version of IMS into the IMS Tools Generic MSC Exit table. To add ITCAM's DFSMSCE0 exit CYMMSC*yz*:

1. Add the following entry into the end of the GEX*xxxx*0 table:

   ```
   EXITDEF(TYPE(MSCE) EXITNAME(CYMMSCyz)
    LOADLIB(HLQ.IMSxxx.SCYMAUTH))
   ```

   where *yz* is the IMS version number (I0 = IMS V8.1, J0 = V9.1, K0 = IMS V10.1, or L0 = IMS V11.1). Follow a similar pattern to add your version of DFSMSCE0, if you are using the user exit.
2. If you are using exit DFSYIOE0, you must still follow ITCAM for Transaction Tracking IMS instructions to link ITCAM IMS exit CYMYIO*yz* to DFSYIOE0 and rename your DFSYIOE0 exit to ZFSYIOE0.

   Note: GEX does not support DFSYIOE0.

For more information, see *IMS Tools Common Services Reference* manual (SC19-2546).

Installing and configuring IMS Connect Tracking

To install and configure IMS Connect Tracking, complete this procedure.

To install and configure IMS Connect Tracking:

1. Ensure SCYMAUTH has been added to the APF list. The SCYMAUTH library must be added to your z/OS APF list.
2. Add SCYMAUTH to the IMS Connect STEPLIB. SCYMAUTH must be added to the IMS Connect STEPLIB DD concatenation before any other libraries.

   Note: The load library for IMS Connect Extensions must be first in the STEPLIB concatenation if installed.
3. Add CYMIHIN DD DSN=hlq.SCYMSAMP(CYM$ICNP),DISP=SHR to the IMS Connect JCL. Only one version of CYM$ICNP is required unless you want to

specify different startup parameters for each IMS Connect region. This DD must point to a valid sequential file with LRECL <=256 holding ITCAM for IMS Connect startup parameters. See the SCYMSAMP CYM$ICNP member for more information on these input parameters.

4. Customize CYM$ICNP. If you customize ITCAM for IMS Connect startup parameters, review and modify the SCYMSAMP CYM$ICNP member. See this member for valid parameters, defaults, and instructions on customizing this file. IMS Connect Tracking is shipped with ICONMON=OFF, so IMS Connect tracking is off at startup unless you change ICONMON=ON.
5. Rename the existing HWSTECL0 Module or run SCYMSAMP(CYMLINK) to linkedit CYMHIN00 as HWSTECL0. IMS Tracking uses the HWSTECL0 exit to perform IMS Connect monitoring. If this exit is already in use, change the name of this module to ZWSTECL0. This module will continue to be called by IMS Connect as if IMS Tracking was not installed. See the SCYMSAMP member CYM$ECL0 for a sample job that renames HWSTECL0.

   Notes:

   - This step does *not* apply to IMS Connect Extensions' HWSTECL0 exit.
   - You can restore your original environment by removing the IMS Tracking libraries from the IMS Connect region's STEPLIB DD and comment out or remove the CYMIHIN DD statement.
6. Restart IMS Connect. Look for the CYM1017I and CYM1012I messages indicating that IMS Tracking initialization has been completed.

Note: IMS Connect Tracking is included with IMS Connect turned off (ICONMON=OFF) in SCYMSAMP member CYMIHIN. To enable IMS Connect Tracking you must either change ICONMON to on (ICONMON=ON) before starting IMS Connect or issue modify command `F CYTAPROC,CYMI` *`imsconnect_jobname`* `ENABLE` as documented in "Operator commands" on page 73.

Using IMS Tracking with WebSphere z/OS Data Collector

If you are using the WebSphere z/OS Data Collector, ensure that you have included the `SCYNAUTH` data set in the WebSphere Application Server Servant Region startup procedure or in `LINKLST`.

See the *ITCAM for WebSphere z/OS Data Collector Configuration and Customization Guide* for more information.

Transaction filtering

Include or exclude IMS transactions from tracking using filtering control statements added to the IMS Tracking or IMS Connect startup PDS members.

IMS Tracking and IMS Connect can share the startup members, or each IMS control region or IMS Connect address space can have its own PDS startup member. Each filtering control statement must have a keyword for the kind of filtering to be done. Asterisks (*) are permissible values; they allow filtering to range over generic keyword values.

There is only one transaction filtering keyword:

`TRANSID=` *`maximum of 8 characters`*

No other filtering keywords are recognized.

All IMS transactions are included for tracking by default, unless they are explicitly excluded by a filtering keyword TRANSID= *value* on an EXCLUDE TYPE=E control statement. EXCLUDE matches can be overwritten by INCLUDE TYPE=I control statements with matching TRANSID= *keyword* values.

Note: INCLUDE TYPE=I control statements have no meaning and have no effect unless they override a matching EXCLUDE TYPE=E filtering control statement.

Examples

The following examples assume that only the control statements in the example can be found in the startup PDS member.

1. The following statement excludes all transactions from tracking:
   ```
   TYPE=E,TRANSID=*
   ```
2. Use these two statements to track all transactions (they are the equivalent of having no filtering control statements in the startup PDS member):
   ```
   TYPE=E,TRANSID=*
   TYPE=I,TRANSID=*
   ```
3. Use these two statements to track all transactions except for those with transaction names beginning with "AB" or "D":
   ```
   TYPE=E,TRANSID=AB*
   TYPE=E,TRANSID=D*
   ```
4. Use these two statements to exclude all transactions except for those with transaction names beginning with "PAR":
   ```
   TYPE=E,TRANSID=*
   TYPE=I,TRANSID=PAR*
   ```
5. Use these two statements to exclude all transactions except for transactions named "PAR":
   ```
   TYPE=E,TRANSID=*
   TYPE=I,TRANSID=PAR
   ```
6. Because TYPE=I entries have no meaning unless there is a match with a TYPE=E entry, these three statements in a PDS member would, by themselves, have no effect on tracking:
   ```
   TYPE=I,TRANSID=ABCD
   TYPE=I,TRANSID=EFG*
   TYPE=I,TRANSID=Y*
   ```

In-Core Filtering Tables

In-core transaction filtering tables are built at IMS control region startup time and at IMS Connect startup time from the filtering control statements in an ITCAM PDS startup member. The in-core filtering tables are located in MVS ECSA in subpool 241, and they are scanned as IMS transactions execute to determine whether a particular transaction is to be tracked. There is at most only one active transaction filtering in-core table for each IMS control region and for each IMS Connect address space.

There is also a dynamic filtering in-core table replacement feature. To use this feature, either modify the existing commands or add the new **TYPE=E** and **TYPE=I** commands to the IMS Tracking or IMS Connect startup PDS member(s) and issue an MVS MODIFY command. The ITCAM MVS Container, which usually runs as job CYTAPROC, can execute MVS operator commands on behalf of IMS Tracking. The **MVS MODIFY** command can be run from a TSO/ISPF session:

```
/F CYTAPROC,CYMI xxxx UFILTERS yyyyyyyy
```

where, *xxxx* is the targeted IMS or IMS Connect id, and *yyyyyyyy* is the IMS Tracking or IMS Connect PDS startup member. When invoked, the **UFILTERS** command replaces the existing transaction filters in-core table for the specified IMS control region or IMS Connect address space with the control statements from the PDS member *yyyyyyyy*. A full replacement of the existing in-core table with a new one is always done when a **UFILTERS** command is executed.

For dynamic filtering table replacements to be done from the ITCAM MVS Container, the CYTAPROC job must have these DD statements pointing to the PDS libraries containing the IMS Tracking or IMS Connect startup members. The startup PDS member names are not present in the DD statements. The startup PDS member names must be supplied on the MVS MODIFY **UFILTERS** commands:

```
//CYMIMSIN DD DISP=SHR,DSN=xxxxxxxx.xxxxxxxx.xxxxxxxx
         *Library with the ITCAM for IMS startup member
//CYMIHIN    DD DISP=SHR,DSN=xxxxxxxx.xxxxxxxx.xxxxxxxx
         *Library with the ITCAM for IMSConnect startup member
```

The filtering library's DD statements in CYTAPROC are needed only if dynamic replacements of in-core filtering tables are to be done.

IMS and IMS Connect Tracking operation

IMS and IMS Connect Tracking provide operator commands by using the Transactions Container. These commands are available if CYMI is specified as the first four characters of the parameters passed to CYTZDRVR, and SCYMAUTH is added to Transactions Container STEPLIB concatenation.

Operator commands

Commands are sent to the Transactions Dispatcher using the z/OS system console command:

```
F proc, CYMI imsid command
```

where *command* is the command to execute and *imsid* is either:

- The four-character IMSID (for IMS regions), or
- The jobname (for IMS Connect)

For example:

- `F CYTAPROC,CYMI IMS1 STATUS` (for IMS), or
- `F CYTAPROC,CYMI IMSCONN1 STATUS` (for IMS Connect)

The available commands are shown in Table 14.

Table 14. IMS and IMS Connect Tracking operation commands

Command	Description
Commands to control IMS and IMS Connect monitoring	
`CYMI imsid DEBUG ON`	Show debug and event information – this option produces a large amount of output. Use this option only if requested to do so by Tivoli Software Support.
`CYMI imsid DEBUG OFF`	Do not show debug information.

Table 14. IMS and IMS Connect Tracking operation commands (continued)

Command	Description
CYMI *imsid* DISABLE	Disable all IMS Tracking monitoring for this IMS subsystem or IMS Connect region. Use the CYMI *imsid* ENABLE command to enable IMS Tracking again.
CYMI *imsid* ENABLE	Enable IMS Tracking for this IMS subsystem or IMS Connect region.
CYMI *imsid* STATUS	Display the status of IMS Tracking monitoring.
Commands to control IMS monitoring	
CYMI *imsid* ADDDATA OFF	Disable adding IMS message data to TTAPI events. Additional data is not sent to the TT Container.
CYMI *imsid* ADDDATA ON	Enable adding IMS message data to TTAPI events. Additional data is sent to TT Container and can be displayed in Container job task SYSOUT by setting the Container DEBUG EVENTS option.
CYMI *imsid* DB2 OFF	Disable all IMS Tracking DB2 monitoring for this IMS subsystem. Use the CYMI *imsid* DB2 ON command to enable IMS Tracking DB2 monitoring again.
CYMI *imsid* DB2 ON	Enable IMS Tracking DB2 monitoring for this IMS subsystem.
CYMI *imsid* SYSPREF OFF	Stop IMS Tracking from using the IMS System Prefix segment. Using this option limits the monitoring that IMS Tracking can perform. Only IMS Connect, IMS OTMA, and Program Routing (PR) tracking is performed. Non IMS Connect and OTMA transactions are not tracked. Use the CYMI *imsid* SYSPREF ON command to allow IMS Tracking to use the IMS System Prefix segment again.
CYMI *imsid* SYSPREF ON	Allow IMS Tracking to use the System Prefix segment. This command enables Transaction Tracking to fully monitor IMS traffic.
CYMI *imsid* TTAPI OFF	Disable Transactions Container support. Events are not sent to the Transactions container.
CYMI *imsid* TTAPI ON	Enable Transactions Container support. Events are sent to the Transactions container.

Chapter 7. MQ Tracking for z/OS

MQ Tracking for z/OS is an extension to the Transaction Tracking for z/OS product, providing support for WebSphere MQ on the z/OS operating system.

MQ Tracking for z/OS consists of the following components:

- Transactions MQ Container: a z/OS Started Task (STC) that provides the container function for MQ events. See "Transactions Base administration and operation" on page 36 further information.
- Transactions MQ Dispatcher: code that runs within the Transactions MQ Container. This code provides an operator command interface, creates queues to receive Transaction Tracking events from MQ subsystems, and manages the MQ Courier task.
- Transactions MQ Courier: code that processes the event queues, validating and forwarding events to the Transaction Collector running on a Windows or UNIX Server.
- Transactions MQ Exits: MQ exit code that is dynamically installed into user-specified MQ environments. This code creates events for MQ activity and sends these events to the MQ Dispatcher using the Transaction Tracking API.

Installing and configuring MQ Tracking for z/OS

Before installing MQ Tracking for z/OS, ensure that you have the required software and understand the installation process.

Software prerequisites

The following software is required:

- z/OS system as described in "Software prerequisites" on page 33
- WebSphere MQ 5.3.1 or later

Installing and configuring MQ Tracking for z/OS

Before installing MQ Tracking for z/OS, ensure that you have the required software installed.

Procedure

To install and configure MQ Tracking for z/OS on your z/OS systems:

1. Install the Transactions Base.

   See "Installing and configuring Transactions Base" on page 34 for details.
2. SMP/E installation.

   Ensure that the MQ Tracking for z/OS component was installed as part of the SMP/E installation (separate FMID HCYT73Q).
3. Post-APPLY link-edit (optional).

   If you are using MQ clusters and have Queue Managers, instrumented for ITCAM for Transactions, connecting to z/OS from distributed systems (via cluster), you must run `SMP/E APPLY` then run the `CYTQBCEX` link-edit job from the `hlq.SCYTSAMP` library. The load module output of the `link-edit` job must be made available in the `MQ CSQXLIB DD` for all cluster-connected channel initiators on z/OS. The sample `JCL CYTQBCEX` contains detailed installation instructions.

4. Determine subsystem names for the MQ container STCs.

   Every Transaction Tracking for z/OS Started Task requires a four-character z/OS subsystem name. Consult with your Systems Programmer for a suitable subsystem name. Although Transaction Tracking dynamically adds this subsystem on startup, you may want to add it to your z/OS IEFSSNxx parmlib member to document its use. This subsystem name is used later in the installation process.

5. Invoke CYTQCACH in hlq.SCYTSAMP to build a cache of WebSphere MQ data to be processed by the CYTQPROC Container STC during initialization.

   a. Modify the dataset names to suit your environment.

   b. Change the WMQ variable to the name of the WebSphere MQ subsystem you plan to track.

   c. Invoke CYTQCACH in hlq.SCYTSAMP.

      Invoke the procedure once for each WebSphere MQ subsystem you want to track. Use DISP=SHR on the first SYSPRINT statement for the first invocation, and DISP=MOD for subsequent invocations.

      The CSQUTIL SYSPRINT dataset must first be allocated with the following attributes:

      ```
      DSORG=PS,RECFM=VBA,LRECL=125,BLKSIZE=27998
      ```

      Note: This JCL may optionally be merged with the CYTQPROC JCL.

6. Copy and customize the CYTQPROC sample procedure to PROCLIB.

   a. Copy the SCYTSAMP member CYTQPROC to a PDS in your z/OS PROCLIB concatenation, renaming it to the name chosen for the Transactions Container STC.

      To enable the WebSphere MQ cache for PUT1 and full shared queue support, modify the CYTQCACH DD statement to point to the cache data set output by the CYTQCACH JCL. The CYTQCACH JCL may be run independently of CYTQPROC or merged into the CYTQPROC JCL. If the JCL is merged, start CYTQPROC after the WebSphere MQ Queue Manager initializes, when CYTQCACH calls the WebSphere MQ utility CSQUTIL to extract WebSphere MQ data. See the CYTQCACH JCL for more details.

   b. Modify the JCL of CYTQPROC to point to your Transaction Tracking datasets. Modify the SSNM variable to the subsystem name chosen earlier. If this subsystem is not defined, Transaction Tracking will dynamically define it on startup.

   c. Optional: Modify the CYTQPROC JCL procedure to include a SYSPRINT DD statement.

      By default, if messages and trace data are requested, the CYTQPROC sends this data to a dynamically allocated DD that is directed to the JES spool. One DD will be allocated for each started Courier (see "Transactions Dispatcher operation" on page 39 for further information about Couriers). However, if a SYSPRINT DD statement is included in CYTQPROC, all messages and trace data is directed to SYSPRINT. SYSPRINT may then be redirected to a data set. This may be useful if JES spool space is limited and you prefer output to be directed to a data set rather than to spool. Ensure that the SYSPRINT data set is sufficiently large to avoid an x37 ABEND.

      For example,

      ```
      //SYSPRINT DD DSN=hlq.SYSPRINT,DCB=(DSORG=PS,LRECL=121), etc
      ```

   d. Ensure that the user ID associated with this STC has read access to all the Transaction Tracking datasets, and an OMVS segment (allowing it to use UNIX Systems Services).

e. Review and customize the SCYTSAMP member CYTQPARM. You will need to update the TTServerString parameter to point to the IP address and port number of the receiving Transaction Collector. This must be in the form: tcp:ip_address:port – for example tcp:10.0.0.1:5455

f. Review the Transaction Tracking commands to be issued on startup in the SCYTSAMP member CYTQCMDS. For example, the **CYTQ INIT CSQ1** command can be used to ensure tracking is automatically instrumented on MQ subsystem CSQ1 when the Container subsystem is started.

g.

7. Ensure that the Transactions Container address space is non-cancellable.

   Update SYS1.PARMLIB member SCHEDxx with the following entries:

   ```
   PPT PGMNAME(CYTZDRVR)
         NOSWAP
         NOCANCEL
   ```

   Use the **SET SCH=xx** operator command to activate the SCHEDxx member without an IPL.

8. Start up Transaction Tracking for z/OS STC.

   a. Start the target MQ subsystems.

   b. Optional: To enable the WebSphere MQ cache for PUT1 and full shared queue support, run the CYTQCACH JCL. The CYTQCACH JCL may be run as a job or as a step at the beginning of the CYTQPROC JCL (see the CYTQCACH and CYTQPROC JCL samples for more details). CYTQCACH builds a cache of WebSphere MQ data that is required for resolution of PUT1 and shared queue activity. CYTQCACH must be rerun whenever changes are made to WebSphere MQ queue definitions.

   c. Issue the START command for the MQ container STC. For example, S CYTQPROC.

9. Validate operation.

   a. Tracking events should be generated for instrumented MQ subsystems, the **CYTQ INIT mq_subsystem** command is used to instrument target MQ subsystems. Debug settings in the **CYTQPARM** member can be used to log event activity to the JES spool associated with the container subsystem.

   b. If the container is correctly configured and connected to a Transaction Collector, MQ related data is visible in the IBM Tivoli Monitoring workspace.

   c. The distributed exit TTDCMqCh is not needed on z/OS by default, but MQ cluster support copies channel definitions around the cluster so the distributed definitions are copied to z/OS. Use the sample JCL in **.SCYTSAMP(CYTQBCEX) to create a dummy exit for any Queue Manager that is clustered with a monitored distributed Queue Manager.

Results

One STC can monitor multiple MQ subsystems, however you may choose to have multiple STCs targeting different MQ subsystem. An MQ subsystem can be instrumented by only one container STC.

For normal operation the CYTQ0000I and CYTQ00010I messages are displayed for all targeted MQ subsystems during container startup.

MQ Tracking for z/OS administration and operation

MQ Tracking for z/OS consists of four components:

- Transactions MQ Container: a z/OS Started Task (STC) that provides basic functionality for the MQ "guest" application, including command processing, message output, and subsystem management. This functionality is provided by the Transactions Base.
- Transactions MQ Dispatcher: code that runs within the Transactions MQ Container that:
  - Creates queues to receive Transaction Tracking events from API callers.
  - Provides utility functions to allow tracing of events and management of queues.

  This functionality is provided by the Transactions Base.
- Transactions MQ Courier: code running within the Transactions Container
  - Validates each event received on these queues, and forwards them to the Transaction Collector on another platform.
  - Provides utility functions to allow tracing of events and management of queues.
- Transactions MQ Exits: code that is dynamically installed into target MQ subsystems that:
  - Collects MQ specific information.
  - Sends MQ specific information to the Transactions MQ Dispatcher using the Transaction Tracking for z/OS API.

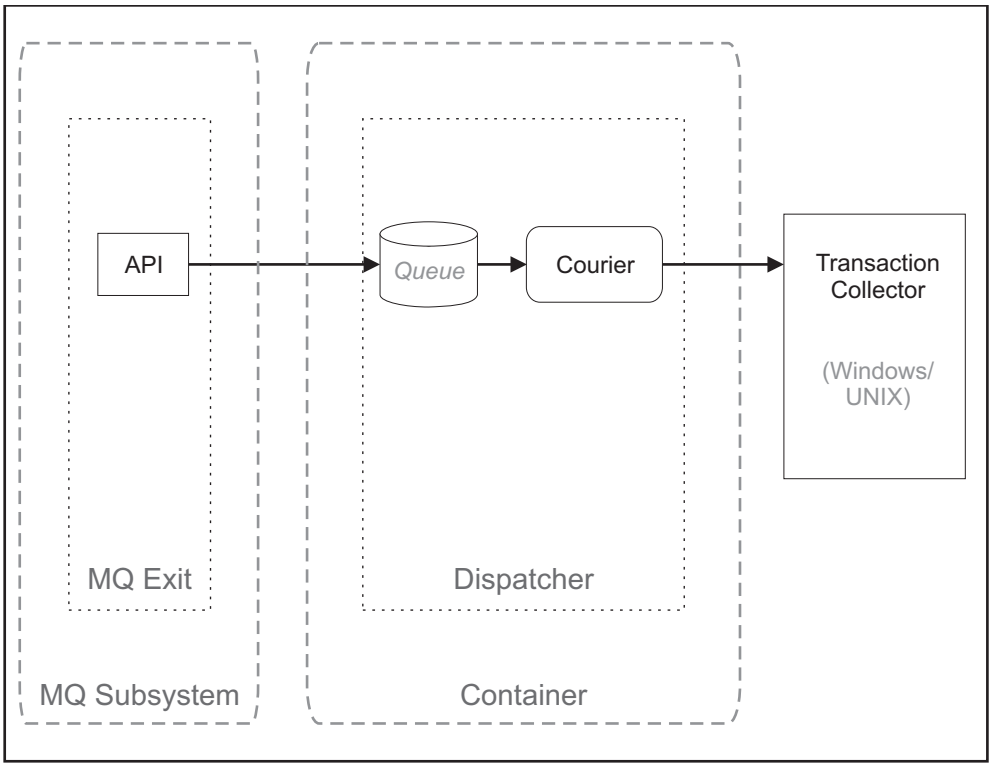


Figure 7. MQ Tracking for z/OS architecture diagram

An operator or configuration file command can be used to install or remove MQ exits into a target MQ subsystem (CYTQ INIT,*mq_subsystem*). After the exits are installed, MQ activity for that subsystem causes events to be generated and sent to the dispatcher.

Transactions MQ Container operation

The Transactions MQ Container is a z/OS Started task inside which all MQ Tracking for z/OS processing is performed.

If you want to enable the WebSphere MQ cache for PUT1 and full shared queue support, start WebSphere MQ and run the CYTQCACH JCL before you start the CYTQPROC STC. CYTQCACH is dependent on the availability of the WebSphere MQ Queue Manager. Alternatively, integrate the CYTQCACH JCL into the CYTQPROC so that the cache data is created when the CYTQPROC is started. Rerun CYTQCACH JCL whenever WebSphere MQ queue definitions change.

Start the Transactions MQ Container using the z/OS system console Start command (eg. S CYTQPROC). Starting the container runs the CYTZDRVR program, which accepts an 8 character input parameter specified in the JCL procedure or overridden on the Start command. The first four characters specify what code will run within the Transactions MQ Container. It must be:

- CYTQ: the WebSphere MQ monitoring code runs inside the container.

The second four characters specify the name of the subsystem that the Transactions MQ Container will use. The Transactions MQ Container will dynamically add this subsystem on startup if it does not already exist.

To stop the Transactions MQ Container, issue the z/OS system console Stop command (eg. P CYTQPROC).

The **CYTQ INIT mq_subsystem** command dynamically installs MQ instrumentation exits into the named MQ subsystem. The **CYTQ TERM mq_subsystem** command removes the exits.

If the target MQ subsystem is not active at the time the command is issued, the command is queued and the exits are installed when the MQ subsystem becomes active. A message is issued to indicate this situation. If the MQ subsystem or the Container address space shutdown at any time after the exits have been installed, the exits are dynamically removed. A message is issued to indicate this situation. If the MQ subsystem needs to be monitored again when it is restarted, issue the command **CYTQ INIT mq_subsystem** manually.

Operator Commands

Commands are issued to the Transactions MQ Container in two ways:

- Using the z/OS system console command F *proc,command*. For example, F CYTQPROC,DEBUG OFF.
- By specifying the command in a sequential data set in the Container CYTACMD DD, described in the CYTQPARM sample. For example, DEBUG OFF.

The available Transactions Container commands are shown in Table 5 on page 39.

Interacting with the RMFT

The MQ Container address space updates the WebSphere MQ RMFT to hook certain MQ function calls. Other products may also update the RMFT, for example, Tivoli OMEGAMON XE for Messaging.

If MQ Tracking for z/OS is not the only RMFT updater, it should be shutdown (or deactivated with the **CYTQ TERM queue_manager** command) in the reverse order to startup (or activated with the **CYTQ INIT queue_manager** command).

For example, if Tivoli OMEGAMON XE for Messaging is started and instruments WebSphere MQ queue manager CSQ1, and then MQ Tracking for z/OS is started and also instruments CSQ1, MQ Tracking for z/OS must be shutdown (or de-instrumented) before Tivoli OMEGAMON XE for Messaging shuts down or de-instruments CSQ1. If you attempt to shutdown (or **CYTQ TERM queue_manager**) in the incorrect order, MQ Tracking for z/OS issues the message `CYTQ0025I` and rejects the shutdown.

Transactions MQ Dispatcher operation

Providing that `CYTQ` is specified as the first four characters of the parameters passed to `CYTZDRVR`, the Transactions MQ Dispatcher code starts up automatically during Transactions MQ Container startup and runs inside the Transactions MQ Container.

Note: No Queues or Event Handlers are started by default. You must issue one Transaction Collector `CYTQ CSTART` command for the Transactions MQ Container via the `CYTQCMDS` member or operator command.

Operator commands

Commands are issued to the Transaction Collector in two ways:

- Using the z/OS system console command `F` *`proc`*`,` `CYTQ` *`command`*. For example, `F,CYTQPROC,CYTQ STATUS ALL`.
- By specifying the command in a sequential data set in the Transactions Container `CYTACMD DD`. For example, `CYTQ STATUS ALL`.

The available commands for the Transactions Dispatcher are shown in Table 6 on page 40.

The available commands for the Transactions MQ Dispatcher are described in Table 15.

Table 15. Transactions MQ Dispatcher operator commands

Command	Description
`CYTQ INIT` *`mq_subsystem`*	Install the Transactions MQ Exits into the specified *mq_subsystem*.
`CYTQ TERM` *`mq_subsystem`*	Uninstall the Transactions MQ Exits from the specified *mq_subsystem*.
`CYTQ VER`	Display version information for the MQ specific code.
`CYTQ STATUS`	Display status information for the MQ specific code.

Table 15. Transactions MQ Dispatcher operator commands (continued)

Command	Description

`CYTQ FILTER mq_subsystem INCLUDE queue_name`
`CYTQ FILTER mq_subsystem EXCLUDE queue_name`
`CYTQ FILTER mq_subsystem CLEAR ALL|INCLUDE|EXCLUDE`
`CYTQ FILTER mq_subsystem DISPLAY ALL|INCLUDE|EXCLUDE`
Note: `mq_subsystem` and `queue_name` can be entered as an asterisk (*), to indicate all queue managers or all queues.
	Add, remove, or display MQ queue filters. See below for details and examples of how to use MQ queue filtering.

Note: Messages and output from the commands in Table 15 on page 80 are displayed in the system console and log.

MQ queue filtering

The **CYTQ FILTER** command can be used to limit the scope of MQ Tracking for z/OS to specific queues. Queues can be included or excluded from processing using operator commands or the CYTQPARM member. Commands are provided to display and clear user entered filters.

Note: Queue filtering can produce unexpected results and possibly break the transaction topology. To link from queue to queue, ITCAM for Transactions requires each adjacent node (queue) to be tracked and queue filtering may remove adjacent nodes. Take special care to filter MCA transmission queues and the `SYSTEM.CLUSTER.TRANSMIT.QUEUE` (in a clustered environment).

Include filters take precedence over exclude filters. Queue names are treated as generic. By default, all queues are included except queues starting with `SYSTEM.` and `AMQ.` (except for the `SYSTEM.CLUSTER.TRANSMIT.QUEUE` which is included).

Examples:

- **Example 1**:

```
F CYTQPROC,CYTQ FILTER MQ61 EXCLUDE * (exclude all queues)
F CYTQPROC,CYTQ FILTER MQ61 INCLUDE PAY (include all queues starting with PAY)
```

- **Example 2**:

```
F CYTQPROC,CYTQ FILTER MQ61 EXCLUDE PAY (exclude all queues starting with PAY)
```

- **Example 3**:

```
F CYTQPROC,CYTQ FILTER * EXCLUDE PAY (exclude all queues starting with PAY
on all currently tracked queue managers)
```

- **Example 4**:

```
F CYTQPROC,CYTQ FILTER MQ61 CLEAR ALL (remove all user entered filters from MQ61)
F CYTQPROC,CYTQ FILTER * CLEAR EXCLUDE (remove all user entered EXCLUDE filters
from all queue managers)
```

- **Example 5**:

```
F CYTQPROC,CYTQ FILTER * DISPLAY ALL (display all user entered filters on all
queue managers)
```

Appendix. Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully.

The major accessibility features in this product enable users to do the following:

- Use assistive technologies, such as screen-reader software and digital speech synthesizer, to hear what is displayed on the screen. Consult the product documentation of the assistive technology for details on using those technologies with this product.
- Operate specific or equivalent features using only the keyboard.
- Magnify what is displayed on the screen.

In addition, the product documentation was modified to include the following features to aid accessibility:

- All documentation is available in both HTML and convertible PDF formats to give the maximum opportunity for users to apply screen-reader software.
- All images in the documentation are provided with alternative text so that users with vision impairments can understand the contents of the images.

Navigating the interface using the keyboard

Standard shortcut and accelerator keys are used by the product and are documented by the operating system. See the documentation provided by your operating system for more information.

Magnifying what is displayed on the screen

You can enlarge information on the product windows using facilities provided by the operating systems on which the product is run. For example, in a Microsoft Windows environment, you can lower the resolution of the screen to enlarge the font sizes of the text on the screen. See the documentation provided by your operating system for more information.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, 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.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Glossary

agent Software installed to monitor systems. The agent collects data about an operating system, a subsystem, or an application.

agent group
A group of management agents that run the same policy or policies. Each management agent is associated with one or more listening and playback components.

agentless
A method a data collection where data is collected from traffic on networks monitored by Web Response Time rather than a domain-specific agent or Data Collector plug-in.

aggregate
(1) An average of all response times detected by the monitoring software over a specific time period. (2) In Transaction Tracking, a node in a transaction topology.

aggregate record
A summary of instance data from all transactions that match a defined pattern.

aggregate topology
A transaction topology that displays all known and implied transactions which may not all be related. See also instance topology.

Aggregation agent
An agent that stores the tracking data from more than one Data Collector plug-in and other monitors and computes aggregates for use by the Transaction Reporter. The Transaction Collector and Web Response Time agent are examples of a Aggregation agent.

aggregation period
The time period, measured in minutes, over which monitoring occurs.

alert A message or other indication that signals an event or an impending event.

application
One or more computer programs or software components that provide a function in direct support of a specific business process or processes.

application pattern
A rule that determines what transactions to monitor and how to group them.

arithmetic expression
A statement that contains values joined together by one or more arithmetic operators and that is processed as a single numeric value. See also arithmetic operator.

arithmetic operator
A symbol, such as + or -, that represents a fundamental mathematical operation. See also arithmetic expression.

ARM-instrumented application
An application in which ARM calls are added to the source code to enable the performance of the application to be monitored by management systems.

attribute
The application properties that are measured and reported on, such as the amount of memory used or a message ID. See also attribute groups.

attribute group
A set of related attributes that can be combined in a data view or a situation.

availability
The successful execution of a monitored transaction over a specified period of time.

client A software program or computer that requests services from a server.

client pattern
A method to define which clients to monitor, and how to group them for reporting.

client time
The time it takes to process and display a web page in a browser.

condition
A test of a situation or state that must be in place for a specific action to occur.

configuration
The manner in which the hardware and software of an information processing system are organized and interconnected.

context
The means used to group tracking data as part of a transaction flow.

Data Collector plug-in
The monitoring component that records the transaction data.

data interval
A time period in minutes for the summary data record. See also summary data.

data source
An application, server, transaction, or other process from which raw data is gathered.

domain
A part of a network that is administered as a unit with a common protocol.

down time
See mean time to recovery.

edge
In transaction monitoring, the point at which a transaction first comes in contact with the monitoring instrumentation.

event An occurrence of significance to a task or system. Events can include completion or failure of an operation, a user action, or the change in state of a process. See also situation.

failure
An individual instance of a transaction that did not complete correctly. See also incident.

firewall
A network configuration, typically both hardware and software, that prevents unauthorized traffic into and out of a secure network.

horizontal
Pertaining to data that is tracked between applications in a domain. See also vertical.

horizontal context
A method of identifying a transaction flow within a transaction which is used to group interactions based on the application supplying the tracking data.

host A computer that is connected to a network and that provides an access point to that network. The host can be a client, a server, or both a client and a server simultaneously.

hot spot
A graphical device used in topologies to highlight the part of an end-to-end transaction that has crossed specified thresholds and has a significant transaction time deviation.

incident
A failure or set of consecutive failures over a period of time without any successful transactions. An incident concerns a period of time when the service was unavailable, down, or not functioning as expected.

instance
A single transaction or subtransaction.

implied node
A node that is assumed to exist and is therefore drawn in the Transaction Tracking topology. An implied node is created when an aggregate collected in an earlier aggregation period is not collected for the current aggregation period.

instance algorithm
A process used by the Transaction Reporter to track composite applications with multiple instances.

instance topology
A transaction topology that displays a specific instance of a single transaction. See also aggregate topology.

interval
The number of seconds that have elapsed between one sample and the next.

linking
In Transaction Tracking, the process of tracking transactions within the same domain or from data collector plugins of the same type.

load time
The time elapsed between the user's request and completion of the web page download.

managed system
A system that is being controlled by a given system management application.

Management Information Base
(1) In the Simple Network Management

Protocol (SNMP), a database of objects that can be queried or set by a network management system. (2) A definition for management information that specifies the information available from a host or gateway and the operations allowed.

mean time between failures
The average time in seconds between the recovery of one incident and the occurrence of the next one.

mean time to recovery
The average number of seconds between an incident and service recovery.

metric A measurement type. Each resource that can be monitored for performance, availability, reliability, and other attributes has one or more metrics about which data can be collected. Sample metrics include the amount of RAM on a PC, the number of help desk calls made by a customer, and the mean time to failure for a hardware device.

metrics aggregation
A process used by the Transaction Collector to summarize tracking data using vertical linking and stitching to associate items for a particular transaction instance. Metrics aggregation ensures that all appropriate tracking data is aggregated.

MIB See Management Information Base.

monitor
An entity that performs measurements to collect data pertaining to the performance, availability, reliability, or other attributes of applications or the systems on which the applications rely. These measurements can be compared to predefined thresholds. If a threshold is exceeded, administrators can be notified, or predefined automated responses can be performed.

monitoring agent
See agent.

monitoring schedule
A schedule that determines on which days and at what times the monitors collect data.

MTBF See mean time between failures.

MTTR
See mean time to recovery.

network time
Time spent transmitting all required data through the network.

node A point in a transaction topology that represents an application, component, or server whose transaction interactions are tracked and aggregated by Transaction Tracking.

over time interval
The number of minutes the software aggregates data before writing out a data point.

parameter
A value or reference passed to a function, command, or program that serves as input or controls actions. The value is supplied by a user or by another program or process.

pattern
A process used to group data into manageable pieces.

platform
The combination of an operating system and hardware that makes up the operating environment in which a program runs.

predefined workspace
A workspace that is included in the software which is optimized to show specific aspects of the collected data, such as agentless data.

probe A monitor that tests a transaction and then detects and reports any errors that were generated during that test.

profile element
An element or monitoring task belonging to a user profile. The profile element defines what is to be monitored and when.

pseudo node
A node that represents an untracked part of a transaction where information about a remote node is provided by a Data Collector plug-in, but that remote node is not itself tracked.

query In a Tivoli environment, a combination of statements that are used to search the configuration repository for systems that meet certain criteria.

regular expression
A set of characters, meta characters, and operators that define a string or group of strings in a search pattern.

reporting rule
A rule that the software uses for naming the collected data that is displayed in the workspaces.

request
See transaction.

response time
The elapsed time between entering an inquiry or request and receiving a response.

round-trip response time
The time it takes to complete the entire page request. Round-trip time includes server time, client, network, and data transfer time.

robotic script
A recording of a typical customer transaction that collects performance data which helps determine whether a transaction is performing as expected and exposes problem areas of the web and application environment.

SAF See Store and Forward.

sample
The data that the product collects for the server.

schedule
A planned process that determines how frequently a situation runs with user-defined start times, stop times, and parameters.

SDK Software Development Kit.

server A software program or a computer that provides services to other software programs or other computers.

server time
The time it takes for a web server to receive a requested transaction, process it, and respond to it.

service
A set of business processes (such as web transactions) that represent business-critical functions that are made available over the internet.

service level agreement
A contract between a customer and a service provider that specifies the expectations for the level of service with respect to availability, performance, and other measurable objectives.

service level classification
A rule that is used by a monitor to evaluate how well a monitored service is performing. The results form the basis for service level agreements (SLAs).

service recovery
The time it takes for the service to recover from being in a failed state.

situation
A set of conditions that, when met, create an event.

SLA See service level agreement.

status The state of a transaction at a particular point in time, such as whether it failed, was successful, or slow.

stitching
The process of tracking transactions between domains or from different types of data collector plugins.

store and forward
The temporary storing of packets, messages, or frames in a data network before they are retransmitted toward their destination.

subtransaction
An individual step (such as a single page request or logging on to a web application) in the overall recorded transaction.

summary data
Details about the response times and volume history, as well as total times and counts of successful transactions for the whole application.

summary interval
The number of hours that data is stored on the agent for display in the Tivoli Data Warehouse workspaces.

summary status
An amount of time in which to collect data on the Tivoli Enterprise Management Agent.

threshold
A customizable value for defining the

acceptable tolerance limits (maximum, minimum, or reference limit) for a transaction, application resource, or system resource. When the measured value of the resource is greater than the maximum value, less than the minimum value, or equal to the reference value, an exception or event is raised.

tracking data
Information emitted by composite applications when a transaction instance occurs.

transaction
An exchange between two programs that carries out an action or produces a result. An example is the entry of a customer's deposit and the update of the customer's balance.

transaction definition
A set of filters and maintenance schedules created in the Application Management Configuration Editor which are applied to the collected data and determine how that data is processed and displayed.

transaction flow
The common path through a composite application taken by similar transaction instances.

transaction interaction
See transaction.

transaction pattern
The pattern for specifying the name of specific transactions to monitor. Patterns define groupings of transactions that map to business applications and business transactions.

trend A series of related measurements that indicates a defined direction or a predictable future result.

uptime
See Mean Time Between Failure.

user profile
For Internet Service Monitoring, an entity such as a department or customer for whom services are being performed.

vertical
Pertaining to data that is tracked within the same application and domain. See also horizontal.

vertical context
The method used to distinguish one transaction flow from another within an application or group of applications. The vertical context enables Transaction Tracking to group individual transactions as part of a flow, label a node in a topology map, and link to an IBM® Tivoli Monitoring application.

view A logical table that is based on data stored in an underlying set of tables. The data returned by a view is determined by a SELECT statement that is run on the underlying tables.

workspace
In Tivoli management applications, the working area of the user interface, excluding the Navigator pane, that displays one or more views pertaining to a particular activity. Predefined workspaces are provided with each Tivoli application, and systems administrators can create customized workspaces.

Index

A

accessibility 83
architecture, Internet Service Monitoring 8

B

books, see publications ix, x

C

CICS
  dynamically-routed transactions, tracking 49
  local transactions, tracking 48
  non-DPL requests, tracking 48
  to DB2 tracking 46
  to IMSDB tracking 47
  umbrella names 49
CICS TG
  configuring to use CICS TG Transaction Tracking 64
  configuring WebSphere Application Server to use CICS TG Transaction Tracking 62
  to CICS tracking 46
  transaction tracking 57
    enabling 61
CICS TG Transaction Tracking
  configuring 59
  installing 59
  properties 60
  software prerequisites 59
CICS Tracking 43
  CICS TG 46
  CICS to DB2 tracking 46
  CICS to IMSDB tracking 47
  configuring 43
  dynamic maintenance 51
  dynamic maintenance, CICS Tracking 51
  functions 53
  installing 43
  programming reference 53
  samples 53
  software prerequisites 43
  TTCU 50
CICS Web Services
  SOAP traffic 47
configuring
  CICS TG Transaction Tracking 59
  CICS Tracking 43
  IMS Tracking 67, 70
  MQ Tracking for z/OS 75
  stand-alone applications to use CICS TG Transaction Tracking 64
  Transaction Tracking for z/OS 33
  Transactions Base on z/OS 34
  WebSphere Application Server to use CICS TG Transaction Tracking 62
conventions, typeface xii

D

Databridge integration 8
Datalog module integration 8
DB2
  to CICS tracking 46
directory names, notation xii

E

enabling
  CICS TG Transaction Tracking on distributed systems 61
environment variables
  notation xii

G

glossary 89

I

IBM Support Assistant xi
  Lite xi
  Log Analyzer xi
IBM Tivoli Monitoring
  component relationships 3
  information center 3
  integration with other products 6
  Internet Service Monitoring integration 8
  ITCAM for Transactions
    framework 15
    product design 15
  overview 3
IMS Connect Tracking
  installing and configuring 70
IMS exits 69
IMS Tools Generic TM 70
IMS Tracking 67
  coexist with IMS exits 69
  installing and configuring 67
  operation 73
  transaction filtering 71
IMSDB
  to CICS tracking 47
installing
  CICS TG Transaction Tracking 59
  CICS TG Transaction Tracking on distributed systems 57
  CICS Tracking 43
  IMS Connect Tracking 70
  IMS Tracking 67
  MQ Tracking for z/OS 75
  Transaction Tracking for z/OS 33
  Transactions Base on z/OS 34
Internet Service Monitoring
  architecture 8
  introduction 8
  monitors
    integration 8
introduction
  Internet Service Monitoring 8
ISA
  See IBM Support Assistant
ITCAM for Transactions
  architecture 17
  component relationships 3
  framework 15
  product 17
  product design 15

L

Log Analyzer xi

M

manuals, see publications ix, x
MQ Tracking for z/OS 75, 78
  installing 75
  installing and configuring 75
  software prerequisites 75
MSC Message Routing Exit 70

N

notation
  environment variables xii
  path names xii
  typeface xii

O

ObjectServer module
  integration 8
online publications, accessing x
operating
  IMS Tracking 73
ordering publications x

P

path names, notation xii
prerequisites
  CICS TG Transaction Tracking 59
  CICS Tracking 43
  MQ Tracking for z/OS 75
  Transaction Tracking for z/OS 33
product
  ITCAM for Transactions
    architecture 17
properties
  CICS TG Transaction Tracking 60
publications ix

publications *(continued)*
accessing online x
ordering x

R

Rational Performance Tester
support 3
Rational Robot
GUI support 3
Response Time
Application Management Console 10
features 10
integration with other products 6
product overview 10
Robotic Response Time agent 10
Web Response Time agent 10
robotic scripts
component relationships 3
RPT
See Rational Performance Tester

S

samples
CICS Tracking 53
support xi

T

Tivoli Data Warehouse
component relationships 3
definition 3
Tivoli Enterprise Monitoring Server
component relationships 3
security 31
z/OS
starting hub 30
Tivoli software information center x
tracking
CICS local transactions 48
dynamically-routed CICS transactions with CICS Tracking 49
non-DPL requests with CICS Tracking 48
using CICS umbrella names 49
transaction filtering
IMS Tracking 71
Transaction Tracking
CICS TG to CICS tracking 46
CICS TG Transaction Tracking properties 60
CICS to DB2 tracking 46
CICS to IMSDB tracking 47
CICS Tracking TTCU commands 50
configuring to use CICS TG Transaction Tracking 64
configuring WebSphere Application Server to use CICS TG Transaction Tracking 62
event 36
for z/OS 33
installing and configuring 33
software prerequisites 33
IMS Connect Tracking installing and configuring 70
Transaction Tracking *(continued)*
IMS Tracking 67
IMS Tracking installing and configuring 67
IMS Tracking operation 73
IMS Trackingtransaction filtering 71
installing CICS TG Transaction Tracking 57
installing for CICS TG 61
main components 13
user interface 13
Transactions Base 36
architecture 36
installing and configuring 34
Transactions Container 36
operation 38
starting 38
Transactions Dispatcher 36, 39
operation 39
Transactions MQ Container 75, 78
operation 79
Transactions MQ Courier 75, 78
Transactions MQ Dispatcher 75, 78
operation 80
Transactions MQ Exits 75, 78
TTCU commands, CICS Tracking 50
typeface conventions xii

V

variables, notation for xii

W

WebSphere z/OS Data Collector 71
workspaces
component relationships 3

Z

z/OS
adding application support 24
adding TT support for TEMS 27
CICS Tracking 43
configuring MQ Tracking for z/OS 75
configuring Transaction Tracking for z/OS 33
enabling Tivoli Enterprise Monitoring Server support 23
IMS Connect Tracking installing and configuring 70
IMS Tracking 67
IMS Tracking installing and configuring 67
IMS Tracking operation 73
IMS Tracking transaction filtering 71
installing and configuring Transactions Base 34
installing MQ Tracking 75
installing MQ Tracking for z/OS 75
installing Transaction Tracking for z/OS 33
MQ tracking 75
MQ tracking software prerequisites 75
z/OS *(continued)*
Tivoli Enterprise Monitoring Server starting hub 30
Transaction Tracking 33
Transaction Tracking for z/OS software prerequisites 33
Transactions Base administration and operation 36
Transactions Container 36, 38
Transactions Dispatcher 36, 39
Transactions MQ Container 79
Transactions MQ Dispatcher 80
uploading data files 23

Printed in USA

SC14-7412-04