IBM Tivoli Composite Application Manager for Transactions

Version 7.3

for AIX, HP-UX, Linux, Solaris, and Windows

Installation and Configuration Guide

SC14-7408-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 523.

This edition applies to V7.3 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

Tables

About this publication

This guide provides information about installing all components in the IBM Tivoli Composite Application Manager for Transactions solution. It also describes the initial configuration required.

Use this information together with the IBM® Tivoli® Monitoring documentation to install and set up your software.

Use the information in the guides listed in Documentation library to understand how to use your new software.

Intended audience

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

Readers should be familiar with the following topics:

- IBM Tivoli Monitoring product
- Tivoli Enterprise Portal interface
- IBM application software

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 Transaction Tracking API User's 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 have some knowledge of 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*

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 this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli software information center website.

Access the Tivoli software information center by going to Tivoli Documentation Central.

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 alphabetical 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 Appendix J, "Accessibility," on page 521.

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.
- 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 and 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.

What's new in ITCAM for Transactions

In this release of ITCAM for Transactions, updates that were included in the previous release are consolidated. New features are described in this section.

Prerequisites

See ITCAM for Transactions V7.3 Prerequisites for the latest information about supported software.

New for all product agents

Component inventory tagging: This release provides a common inventory reporting and collection function that automatically collects information about your installation. This information includes details about the product components that are installed, and version information. This information, based on special inventory tagging files that are included during product installation, is available to IBM support personnel. As a result, you no longer need to provide this information each time you submit a problem report. This feature helps speed resolution of customer problems, reduces service errors, and enhances customer satisfaction.

Integration with IBM Support Assistant: This release provides additional integrated capability with IBM Support Assistant version 4 with Transaction Tracking and Response Time agents. Internet Service Monitoring agent support is not available. You can now run a script to collect certain log files and configuration files in to a compressed package. You can then include this package as part of a PMR that you open and submit to IBM Software Support. Online instructions in the script guide you to complete the collection of information needed to help resolve your problem. See the *Troubleshooting Guide* for more information about working with the **runISAlite** script.

Tivoli Common Reporting version 2.1: Support for Tivoli Common Reporting has been updated to support version 2.1, for both Cognos and BIRT reports. See *Appendix B* in the *Administrators Guide* for more information.

Agentless Transaction Tracking installation option: The ITCAM for Transactions installer now offers a new selection option to install the Web Response Time agent and Transaction Reporter together for TCP transaction tracking. See Chapter 3 in the *Installation and Configuration Guide*.

Online User Assistance updated: The online help system has been updated with the latest information about attributes, workspaces, situations, and Take Action commands for this release.

Application Management

The Application Management Configuration Editor has the following enhancements:

- **Components feature:** Create and manage components for the TCP transaction traffic that you want to monitor. Use the Components feature of Application Management Configuration Editor to define the components and protocols to be monitored. You can use this feature to provide a descriptive reporting name for all TCP traffic associated with a given components entry.

- **Backup and Restore of Application Management Configuration Editor settings:** A procedure for backing up and restoring configuration settings in the Application Management Configuration Editor is added to the *Administrators Guide*.

Internet Service Monitoring updates

The Internet Service Monitoring component of ITCAM for Transactions has the following updates:

- **Internet Service Monitoring Configuration command-line interface**: A new interface which provides the same functionality as the Internet Service Monitoring Configuration user interface. The Internet Service Monitoring Configuration command-line interface incorporates the functionality of **ismbatch** in to the seamless integration with the Internet Service Monitoring Configuration user interface, and updating the Tivoli Enterprise Portal Server database.
- Internet Service Monitoring monitors are now supported on Linux on System z.

Response Time updates

The Response Time component of ITCAM for Transactions has the following updates:

- **Web Response Time Agentless Transaction Tracking:** This release of ITCAM for Transactions introduces a new way to track transactions without the need for domain specific or application-specific data collectors. This type of monitoring is called *Agentless Transaction Tracking*.

  Agentless Transaction Tracking extends the capabilities of existing ITCAM for Transactions features and function:

  – The Web Response Time agent is used to monitor generic TCP/IP based network flows.
  – Enhanced Tivoli Enterprise Portal workspaces are provided with 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 Transaction Collector, 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. You can then 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. The Web Response Time agent then applies these configuration changes to the monitored Internet Protocol network flow data.

  This approach offers you greater flexibility in configuring tables and topologies, along with quicker time to value and end to end tracking ability.
- **New Web Response Time workspaces:** Several new workspaces are provided in this release to display data collected by Agentless TCP transaction tracking. The Components workspace replaces the previous default workspace displayed on the Network node of the Navigator view. This workspace focuses on data aggregated by unique components (such as IBM HTTP Server, WebSphere Application Server, and others) monitored by the agent.

  Additional linked workspaces provide more details for a selected component:

  – Client Facing Components workspace

  - Component Details workspace
  - Component Server Details workspace
  - Server Dependencies workspace
  - Client Dependencies workspace
  - Component History workspace
- **Components and Protocols in Application Management Configuration Editor:** New capability is added to the Application Management Configuration Editor to create and manage components and protocols used with Agentless Transaction Tracking. See the *Administrators Guide* for more information.
- **Web Response Time agent configuration changes:** The configuration options for the Web Response Time agent are reorganized into several new dialog windows. These windows include new options for enabling and monitoring agentless TCP transaction tracking. The configuration procedure is now segmented into basic and advanced configuration options. To help limit and refine the amount of TCP transaction tracking data collected, you can define one or more network masks to exclude certain data from being monitored. You can also define one or more server masks that ensure that TCP data is displayed properly in topology views. See the *Installation and Configuration Guide* for more information about configuring agents.
- **WRT TCP Status attribute group:** A new Web Response Time attribute group, WRT TCP Status, is included in this release. This group contains attributes used in the Agentless Transaction Tracking workspaces and views. See the *Users Guide* for more information about this and other ITCAM for Transactions attribute groups.
- **Socket-based script playback:** With the availability of agentless transaction monitoring of communication between servers, you can use socket-based playback of Rational Performance Tester scripts to capture TCP tracking data. This capability offers a powerful playback solution for client applications (such as Internet Explorer, applet, and socket applications). For more information, see the *Administrators Guide*.
- **New sit2profile command-line option:** A new command-line option (**-s**) is added to the **sit2profile** tool. See the related appendix in the *Installation and Configuration Guide* for more information.
- **Tech Notes Incorporated into Documentation:** Over 40 tech notes from the IBM Support Portal are incorporated into the documentation for this release. Many of these tech notes are new topics in the *Troubleshooting Guide*, and others providing more information in various areas of the documentation.

Transaction Tracking updates

The Transaction Tracking component of ITCAM for Transactions has the following updates:

- **Agentless transaction tracking**: Provides a new way to monitor your enterprise. Agentless tracking uses data derived from TCP traffic on networks monitored by the Web Response Time agent to detect protocols and applications. New Transactions Overview and Agentless Data workspaces in Transaction Tracking display this data in topologies and tables. When combined with agent-based data, you can link from the overview information to in-depth information about your system.
- **.NET Data Collector**: Major updates to .NET Tracking, which is now called .NET Data Collector. .NET Data Collector supports tracking LDAP, SOAP transactions for Web Services, and ADO.NET. .NET Data Collector also incorporates the functionality provided in earlier releases by IIS Tracking.

- **Data Collector for WebSphere Message Broker**: A new converged WebSphere Message Broker data collector replaces the following existing data collectors:
  - ITCAM for SOA Message Broker data collector from ITCAM for SOA
  - WebSphere Message Broker Tracking from ITCAM for Transactions

  The new Data Collector for WebSphere Message Broker can be installed with either product.
- **For z/OS**:
  - Extended support for WebSphere MQ shared channels, shared queues, and Queue Sharing Groups.
  - Extended support for PUT1 to alias, remote and shared queues.
  - Updates to the CYTQPROC sample JCL for the Container STC and a new JCL sample CYTQCACH has been added. Review these two members in SCYTSAMP if you require shared channel, shared queue, Queue Sharing Group, or PUT1 support.

    The new support uses a cache of WebSphere MQ data that is extracted by using the **CSQUTIL** utility. Build the cache while WebSphere MQ is available and before the Container STC is started. The cache is built by using the **CYTQCACH**sample JCL which you can run as a stand-alone job or as a step in the **CYTQPROC** JCL.

    The new cache facility might require periodic recycles of **CYTQPROC**. Changes to WebSphere MQ made after **CYTQPROC** starts are not reflected until the cache data set is refreshed and **CYTQPROC** is restarted. Whenever WebSphere MQ alias queues, shared queues, or remote queues are defined, update the cache, and restart **CYTQPROC**.

Chapter 1. ITCAM for Transactions introduction

IBM Tivoli Composite Application Manager for Transactions (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 integrates several components for measuring internet services, response times, and tracking transactions, and performing deep dive analyses so you can identify and solve problems.

ITCAM for Transactions includes the following components:

- Internet Service Monitoring
- Response Time
- Transaction Tracking

ITCAM for Transactions 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 provides a solution to rapidly isolate problem components and speed 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.
- Integrates several capabilities for measuring internet services, response times, tracking transactions, and performing deep dive analyses so you can identify and solve problems. Using the single dashboard interface, you can detect a problem when it occurs or even *before* it occurs, pinpoint the problem to a specific part of your IT environment, and then hand off the details to the appropriate specialist to do the deep dive analysis and 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 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).

  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
  - Client Response Time
  - 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
  - Data Collector for WebSphere Message Broker
  - MQ Tracking
  - .NET Data Collector
  - Tuxedo Tracking
  - WASTT
  - ITCAM for SOA Log File Service
  - 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:
- Web Response Time
- WebSphere Application Server
- IBM HTTP Server
- ITCAM for Application Diagnostics
- ITCAM for J2EE
- ITCAM for SOA
- Optim Performance Manager
- 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 14 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.

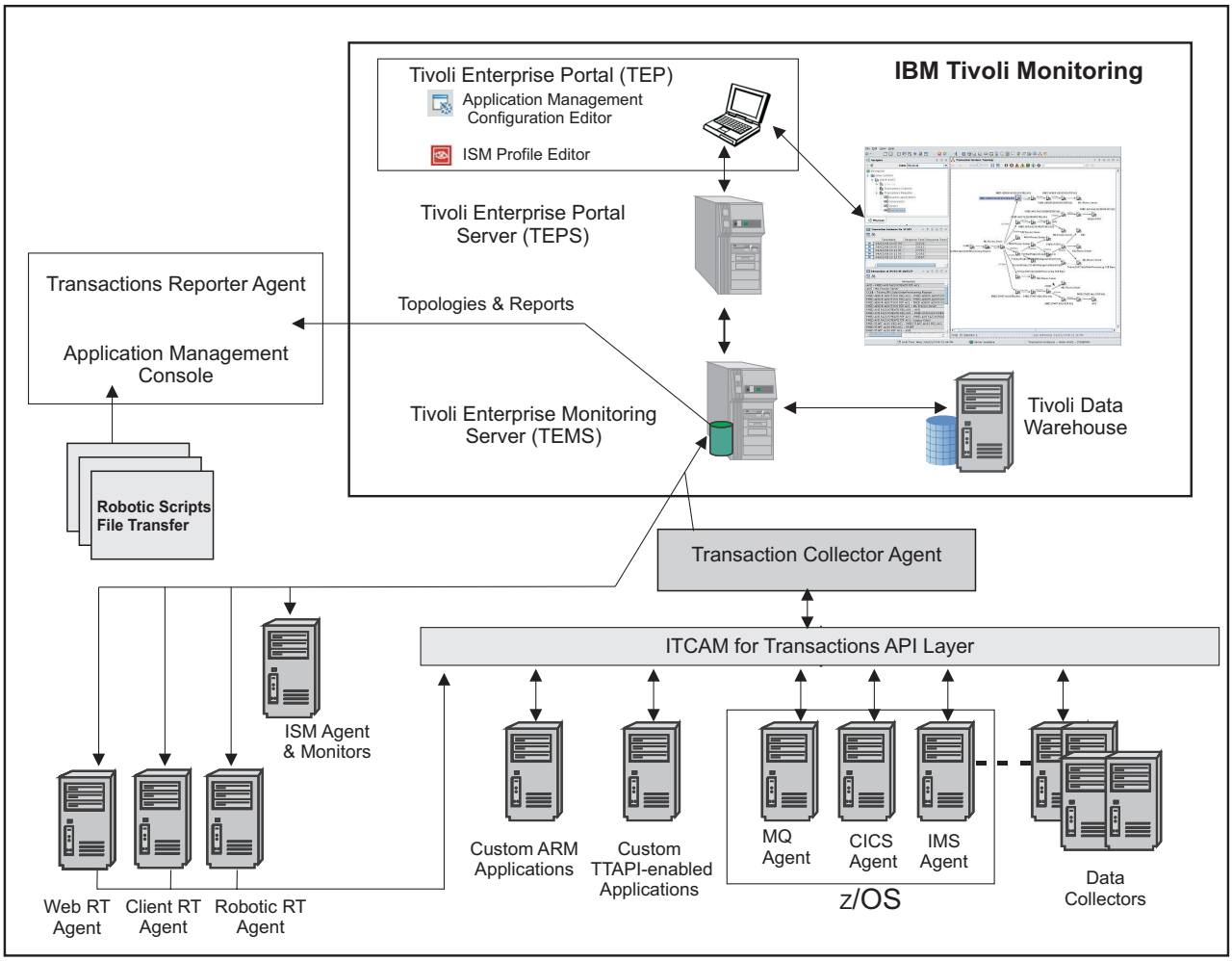


Figure 1. How ITCAM for Transactions integrates with IBM Tivoli Monitoring

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 2 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.

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

Component	Description
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.
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. • Client Response Time - reports real-user response time of Microsoft Windows applications at the client levels that can be broken down into overall response time, server time, and network time. This agent is ideal for monitoring client applications such as Lotus Notes® and Microsoft Outlook. • 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. • ITCAM for SOA Log File Service - gathers monitoring information collected in IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA) log files and converts it to a format suitable for display in Transaction Tracking workspaces and views.

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

Component	Description
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.

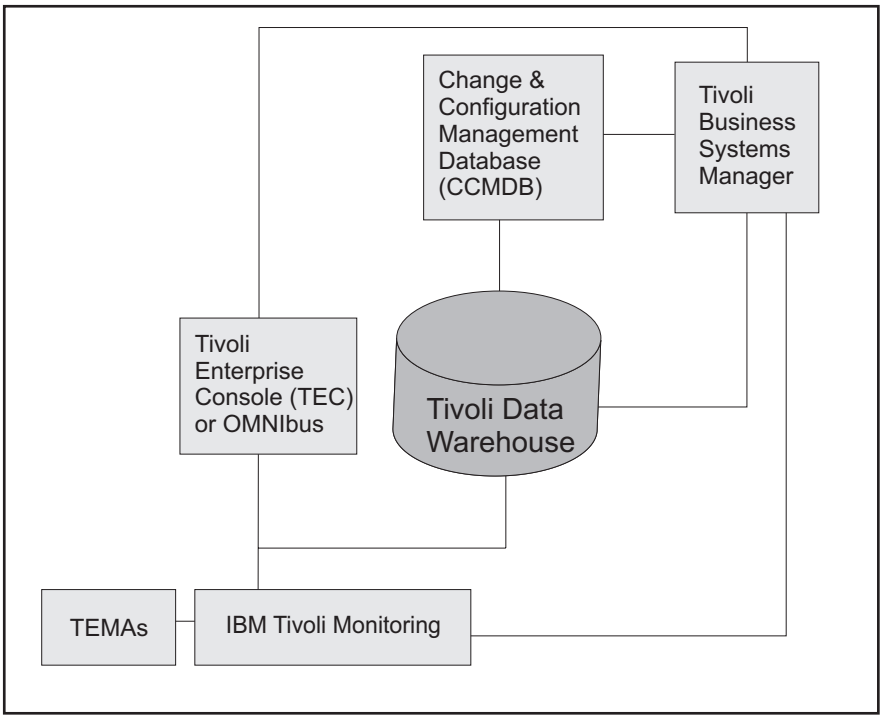


Figure 2. Integration of IBM Tivoli Monitoring and other products

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.

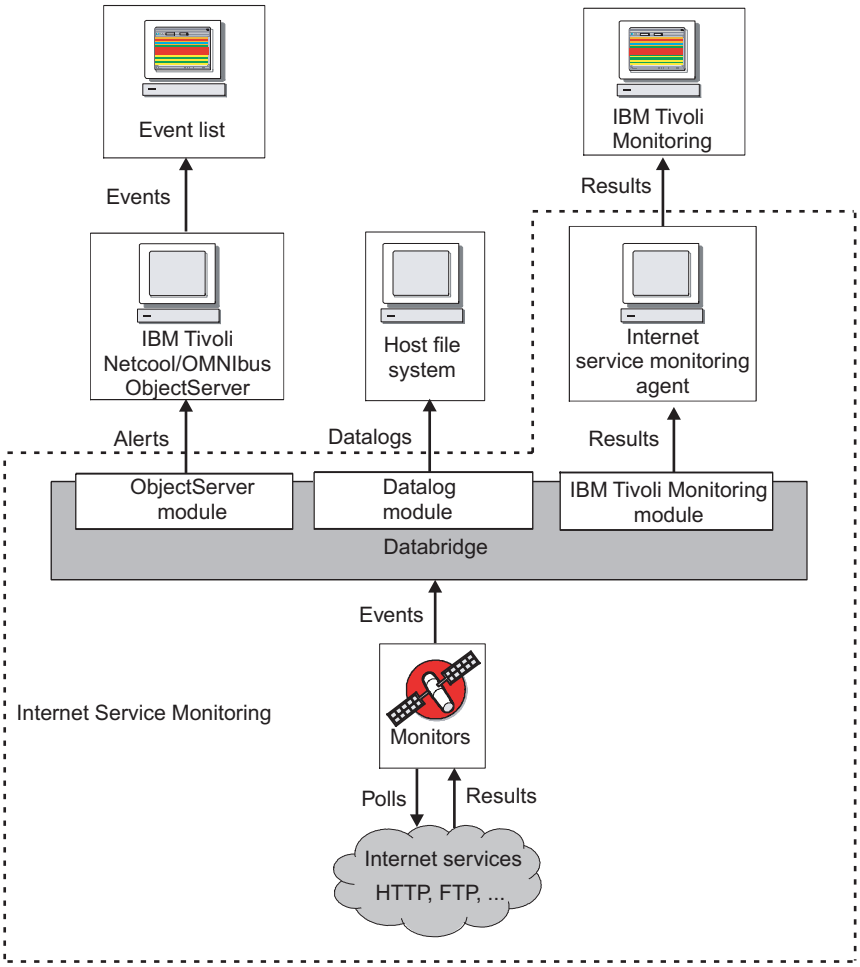


Figure 3. Internet Service Monitoring architecture

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
- Client Response Time
- 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 TimeResponse 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.

Client Response Time

Provides real end-user client monitoring for Microsoft Windows desktop applications. It is installed separately on client desktop systems. Client Response Time provides the following features:

- Measures and collects application transaction response time on Windows system clients from the end-user perspective. For example, you can use Client Response Time to monitor Lotus® Notes® response times, ensuring that Lotus Notes responds quickly to support demands for corporate productivity.
- Gathers data for Windows system clients from real end-users for reporting on Service Level Agreements (SLAs).
- Monitors virtually any Windows system application with a graphical user interface, when you create custom behavior files with the ETEWatch® Customizer.
- Reports the details of client, network, and server response times.
- Supports all ETEWatch behavior files and Response Time CAT behavior files.
- Supports applications running on Citrix and Terminal servers.
- Provides support for:
  - Lotus Notes
  - Microsoft Outlook
  - SAP GUI
  - IBM PCOMM (TN3270 protocol only)
  - Hummingbird (TN3270 protocol only)
  - Exceed (TN3270 protocol only)
  - Attachmate Extra TN3270 emulators

  The supported applications require no special configuration after installation.

Client Response Time also provides the following features:

- Custom Application Response Measurement (ARM) monitoring for Client Response Time. ARM is an application programming interface (API) that is used to monitor the availability and performance of business transactions within and across diverse applications and systems.
- Real end-user response time and availability monitoring for custom ARM applications by using the Generic ARM component of Client Response Time.

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. It performs this monitoring by using robotic technology to record and play back transactions that 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.
- Provide HTTPS monitoring by using server plugins for web servers that use temporal forms of encryption for:
  - IBM HTTP Server
  - Apache HTTP Server
  - Internet Information Services
  - Sun Java Web Server (iPlanet)
- By default, the Web Response Time agent monitors all network interfaces. However, it can also monitor a specific network interface. By default, the 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 enables you to observe transactions across products, providing easier integration between different products.

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. The product 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 historic 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 (was ITCAM for WebSphere). 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 17 illustrates this design.

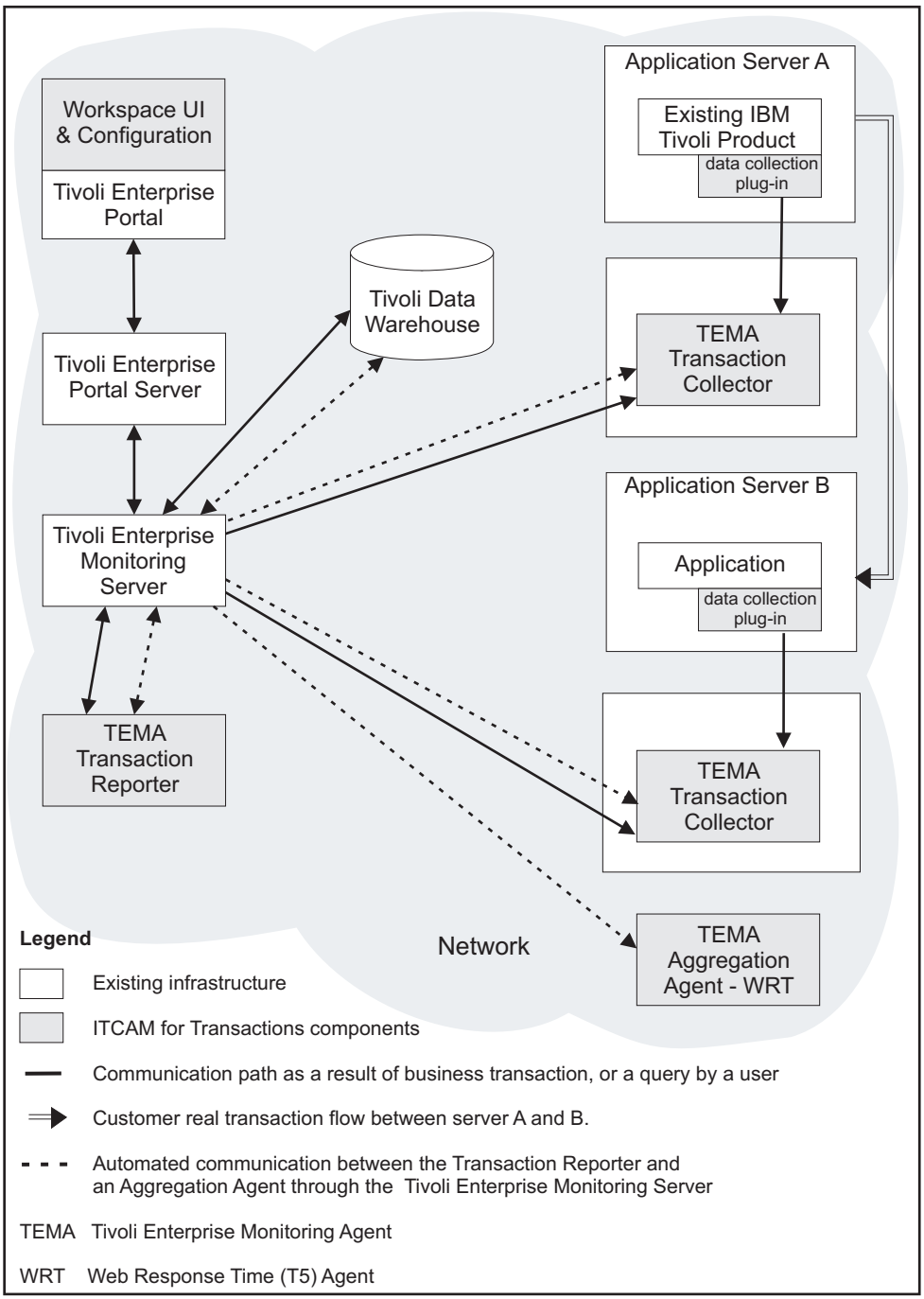


Figure 4. How Transaction Tracking fits in to IBM Tivoli Monitoring

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.

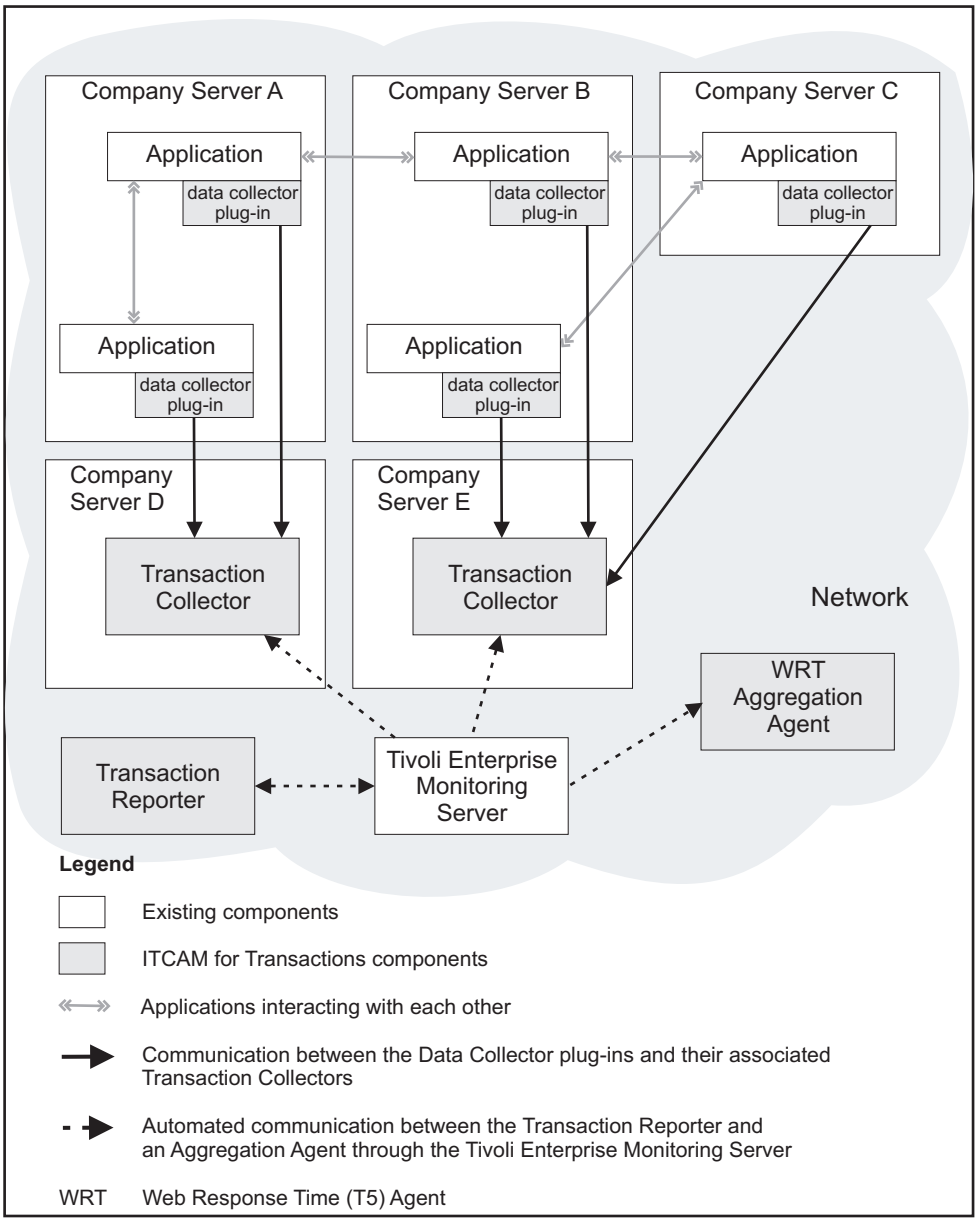


Figure 5. IBM Tivoli Composite Application Manager for Transactions component interaction diagram

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.
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.
.NET Data Collector	Tracks: • ADO.NET interfaces • IIS Server basic authentication • SOAP transactions through .NET ASMX Web Services (also known as ASP.NET web services) • SOAP transactions through Windows Communication Foundation (WCF) services
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.

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

Data Collector plug-in	Description
ITCAM for SOA Log File Service	Gathers monitoring information collected in IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA) log files and converts it to a format suitable for display in Transaction Tracking workspaces and views.
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 Client Response Time and 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. Planning your installation

Before installing ITCAM for Transactions, you must plan what you want to monitor, which components to use, and where to install these components in your unique environment.

This section provides general information. For further information, see ITCAM for Transactions Planning and Deployment Best Practices.

Setting up an effective monitoring environment

To set up an effective monitoring environment, you need to choose the correct agents with which to monitor, the correct robotic playback component, and determine naming conventions that must be followed.

Choosing which Tivoli Enterprise Management Agent to use

To set up an effective monitoring environment, you need to choose the correct agents with which to monitor, the correct robotic playback component, and determine naming conventions that must be followed.

If you want to	Use this agent
View a consolidated enterprise view of all application performance & availability. Create custom roles to limit access to application data.	Application Management Console
Monitor real end-user response times for Lotus Notes, Microsoft Outlook, or applications running in a Citrix or Terminal Services environment. Understand real user (Internal) client experience. Monitor custom Windows applications. Monitor custom ARM enabled applications for clients or desktops. Monitor real user 3270 transactions.	Client Response Time
Determine whether a particular internet service is performing adequately, identify problem areas, and report service performance measured against service level agreements by emulating the actions of a real user. Services that you can monitor include DHCP, DNS, FTP, HTTP, HTTPS, ICMP, IMAP4, LDAP, NNTP, NTP, POP3, RADIUS, RPING, RTSP, SAA, SIP, SMTP, SNMP, SOAP, TCPPort, and TFTP.	Internet Service Monitoring
Run an existing Mercury LoadRunner script and monitor the results with this product. Run robotic monitoring for web applications, Siebel, SAP, Citrix. Run a custom application, script, or command and see results. For example: • Testing server availability with FTP, telnet, or ping • Querying a database with a custom SQL command • Running a custom shell script	Robotic Response Time

If you want to	Use this agent
Display agentless data, such as topology and metric information, derived from TCP traffic on your network monitored by the Web Response Time agent. Requires both Web Response Time and Transaction Tracking. Monitor transactions within and between ARM-enabled applications or implement Transaction Tracking API in your own application to track transactions and determine possible problems. Applications that you can monitor by using ARM or Transaction Tracking API include: • WebSphere Application Server, IBM HTTP Server, DB2, and J2EE servers • WebSphere MQ on distributed and z/OS platforms • Tuxedo • CICS on z/OS • IMS on z/OS Integrate with other products such as the following, to provide end-to-end transaction tracking: • ITCAM for Application Diagnostics • ITCAM for J2EE • ITCAM for SOA • Tivoli Business Service Manager • Robotic Response Time • Web Response Time • Optim Performance Manager • Tivoli OMEGAMON XE for CICS • Tivoli OMEGAMON XE for IMS • Tivoli OMEGAMON XE for Messaging • Monitoring Agent for Microsoft .NET Framework • Monitoring Agent for Microsoft Internet Information Services • Monitoring Agent for Active Directory	Transaction Tracking
Detect agentless data by using TCP traffic in your enterprise, such as protocols and applications. Requires both Web Response Time and Transaction Tracking. Monitor real user transactions for an HTTP server, such as the following measurements: • The time it takes the web server to process and respond to the HTTP request. • The time it takes to display a web page in a web browser. • The time it takes to complete the entire page request (round-trip time). This measurement includes the previous 2 times and network and data transfer time. Discover new URLs.	Web Response Time

Choosing the correct robotic playback component

Use the following table to select the correct robotic playback component for your environment.

Use → If you want to monitor ↓	Rational® Performance Tester	Robot GUI or Rational Functional Tester	CLI Command	LoadRunner	Robot VU *
Web (HTTP or HTTPS)	●				●
Windows GUI clients		●			
CLI commands			●		
Mercury LoadRunner				●	
Citrix	●				
SAP	●				
Siebel	●				
* Robot VU is provided for legacy support. Use Rational Performance Tester for HTTP support.					

Naming conventions for applications

The Tivoli Enterprise Portal has 26 character limitation for the host and application names in the Navigator. When you have a combination of names longer than 26 characters, the application name is truncated. For example, if you have application names such as `Websphere Plants` and `Websphere Petstore` on a computer with a long hostname, the names might appear in Navigator as `WebspherePl` and `WebspherePe`. If the hostname is only four characters shorter, the names are `WebspherePlants` and `WebspherePetsto`.

Install Application Management Console on a computer with a short host name to avoid this limitation.

Planning ITCAM for Transactions deployment and installation

To display ITCAM for Transactions monitoring data in the Tivoli Enterprise Portal, you first install the ITCAM for Transactions components and application support files.

Before you install ITCAM for Transactions, you must first plan the distribution of the Internet Service Monitoring, Response Time, and Transaction Tracking components. After you have installed the ITCAM for Transactions components, you can install their subcomponents. The naming of the subcomponents and the way in which they are installed is unique to each component:

- Internet Service Monitoring *monitors* are installed by the Internet Service Monitoring installer.
- Response Time *monitoring agents* T3 - T6 are installed separately by the Response Time installer.

- Transaction Tracking *Data Collector plug-ins* are optional and, except for MQ Tracking and Data Collector for WebSphere Message Broker, are installed separately with their own installers.

You can install and upgrade ITCAM for Transactions components by using the ITCAM for Transactions installer or the individual installers for each component.

Install a component *either* on a computer by itself *or* on a computer with an installed portal server, monitoring server, or portal client.

Table 5. Deployment and installation checklist

What to do	Where to find more information
Obtain the current architecture diagram (detailed topology) for your environment.	Work with your domain and group owners to create a comprehensive diagram of your transaction environment.
Correlate the current architecture with the platform support matrix. You can monitor only supported platforms and applications.	For further information, see the Prerequisites pages. Select the required version of ITCAM for Transactions on Documentation Central. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**. Select the required component in **Supported operating systems** or **Compatible software**.
Create an application deployment diagram for the distribution of ITCAM for Transactions components.	Using your architecture diagram as a base, determine the following information: • Where to install the IBM Tivoli Monitoring components • Number and placement of individual ITCAM for Transactions components • Number, placement, and type of monitoring agents and Data Collector plug-ins required See "Deploying ITCAM for Transactions" on page 28 for further information.
Create a composite application interaction diagram	The composite application interaction diagram can help you to understand the target application communication paths between its constituent components. Use this diagram to list the communication protocols between its components, and extract expected traffic volume from the business application perspective. Use this diagram to perform an installation validation, and confirm that ITCAM for Transactions can be used to track all stages of the target application, based on communication protocols and domains involved. This diagram also helps you calculate the expected traffic at each Data Collector plug-in.
Determine network considerations that might affect the architecture	Consider the following factors: • Firewalls • Network segmentation • Network speed and traffic
Estimate the sizing requirements to handle the load	Contact your IBM representative for further information.
Complete the Warehouse load projection spreadsheet	See Tivoli Monitoring 6.x Warehouse Load Projections spreadsheet.
Determine high availability and disaster recovery requirements	See IBM Tivoli Monitoring Deployment Guide.
Ensure that you have the necessary IBM Tivoli Monitoring infrastructure	See Table 6 on page 29.

Table 5. Deployment and installation checklist (continued)

What to do	Where to find more information
Download the required software for your supported operating systems including all components and Data Collector plug-ins	Either download the software from Passport Advantage, or use the product installation media. For further information, select the required version of ITCAM for Transactions on Documentation Central. In the navigation pane, select **Composite Application Manager for Transactions** > **Download information**.
Verify that the software and hardware requirements for the components or Data Collector plug-ins that you want to install have been met.	See "Prerequisites" on page 32.
Collect the necessary information for the installation and configuration.	See "Information to collect before you begin installation and configuration" on page 32.
Install the required components by using the ITCAM for Transactions installer or the installers for each component.	See the following resources for the ITCAM for Transactions installer and each component: • Chapter 3, "Installing ITCAM for Transactions by using the ITCAM for Transactions installer," on page 47 • Chapter 4, "Installing Internet Service Monitoring," on page 55 • "Installing Response Time monitoring agents and related software" on page 101 • Chapter 8, "Installing Transaction Tracking," on page 187
Install agent-specific application support.	See the following resources for each component: • Chapter 4, "Installing Internet Service Monitoring," on page 55 • "Installing Response Time monitoring agents and related software" on page 101 • Chapter 8, "Installing Transaction Tracking," on page 187
Configure the components.	See the following resources for each component: • "Configuring components by using the ITCAM for Transactions installer" on page 53 • Chapter 5, "Configuring Internet Service Monitoring," on page 83 • Configuring Response Time and related software • Chapter 9, "Configuring Transaction Tracking," on page 217
Verify the component installation.	See the following resources for each component: • "Troubleshooting the Internet Service Monitoring installation" on page 77 • "Verify installations of Response Time monitoring agents" on page 125 • "Verifying the Transaction Tracking installation" on page 208
Configure the Eclipse help server if required.	The Eclipse help server is updated with Tivoli Enterprise Portal Server application support. If you need to manually update the Eclipse help server, see Chapter 12, "Configuring the Eclipse help server," on page 321 for further information.

Table 5. Deployment and installation checklist (continued)

What to do	Where to find more information
Install and configure the required monitors, monitoring agents, and Data Collector plug-ins.	• For Internet Service Monitoring, see Chapter 4, "Installing Internet Service Monitoring," on page 55. • For Response Time, see Chapter 6, "Installing Response Time," on page 101. • For Transaction Tracking, see Chapter 10, "Installing and configuring Transaction Tracking Data Collector plug-ins," on page 221.
Set up remote deployment if required. *(Optional)* ITCAM for Transactions supports remotely deploying the components across your environment from a central location, the Tivoli Enterprise Monitoring Server.	See Chapter 11, "Working remotely," on page 309.
Configure IBM Tivoli Monitoring to forward events to Tivoli Enterprise Console if required. *(Optional)*	See Appendix E, "Tivoli Enterprise Console event mapping," on page 349.
Start the components.	See "Starting and stopping monitoring agents" on page 326.
Configure historical data collection.	See Setting up historical data collection in the *Administrator's Guide*.
Install a language pack. *(Optional)* You can install local language support on each computer on which the Tivoli Enterprise Portal is located	See Appendix A, "Installing and uninstalling the language pack," on page 329.

Deploying ITCAM for Transactions

To optimize the performance of ITCAM for Transactions, plan the deployment of the components, subcomponents, monitoring agents and Data Collector plug-ins.

Before installing ITCAM for Transactions, using the architecture diagram for your environment as a base, create an application deployment diagram to decide where to locate the ITCAM for Transactions components, subcomponents, monitoring agents and Data Collector plug-ins and the base IBM Tivoli Monitoring software. Factors such as firewalls, network segmentation, and network speed all affect your deployment.

Deploying IBM Tivoli Monitoring components

Plan your IBM Tivoli Monitoring installation after considering the information in Table 6 on page 29.

Table 6. Locating IBM Tivoli Monitoring components

Component	Possible location
Tivoli Enterprise Monitoring Server	The Tivoli Enterprise Monitoring Server is the collection and control point for many ITCAM for Transactions functions, including: • Receiving alerts from agents. • Collecting performance and availability data from the agents. • Tracking the status of monitoring agents. • Distributing situations to the monitoring agents. Install the hub Tivoli Enterprise Monitoring Server inside the environment on a high performance network. Ensure that the connectivity between the Tivoli Enterprise Portal Server and the hub monitoring server, and between the hub monitoring server and the remote monitoring servers is fast and reliable. Install the remote Tivoli Enterprise Monitoring Server taking into account the following information: • Plan your firewalls to ensure that communications can be established between the IBM Tivoli Monitoring components with as few holes in the firewall as possible. • If using many ITCAM for Transactions agents, locate the remote Tivoli Enterprise Monitoring Server inside the DMZ together with the agents to reduce the number of holes in the firewall.
Tivoli Enterprise Portal Server	The Tivoli Enterprise Portal Server maintains a persistent connection to the hub Tivoli Enterprise Monitoring Server. To provide optimal Tivoli Enterprise Portal performance, locate the Tivoli Enterprise Portal Server in the same LAN segment as the hub monitoring server.
Warehouse Proxy Agent	The Warehouse Proxy Agent writes the historical data to a supported relational database. Install the Warehouse Proxy Agent on the same LAN segment as the Tivoli Data Warehouse database server to minimize network transmission delay during processing. If the Warehouse Proxy Agent is placed on a separate server, ensure that it connects to the Tivoli Data Warehouse database server with a high-speed network connection. If you install both the Warehouse Summarization and Pruning Agent and the Warehouse Proxy Agent on the Tivoli Data Warehouse database server, performance can be adversely affected. For agents that connect to a (Network Address Translation (NAT) network, place the Warehouse Proxy Agenton the same server as the remote Tivoli Enterprise Monitoring Server.
Warehouse Summarization and Pruning Agent	The Warehouse Summarization and Pruning Agent aggregates and prunes the ITCAM for Transactions historical data in the Tivoli Data Warehouse. Install the Warehouse Summarization and Pruning Agent on the Tivoli Data Warehouse database server to minimize network transmission delay during processing. If the Warehouse Summarization and Pruning Agent is placed on a separate server, ensure that it connects to the Tivoli Data Warehouse database server with a high-speed network connection. If you install both the Warehouse Summarization and Pruning Agent and the Warehouse Proxy Agent on the Tivoli Data Warehouse database server, performance might be adversely affected.

See IBM Tivoli Monitoring Deployment Guide for further information.

Deploying ITCAM for Transactions components

Plan your IBM Tivoli Monitoring installation after considering the information in Table 7.

Table 7. Locating ITCAM for Transactions components

Component	Subcomponent	Possible location
Internet Service Monitoring See "Planning deployment" on page 34 and "Deployment scenarios" on page 35 for further information.		Install onto a stand-alone system, onto the same system as the IBM Tivoli Monitoring environment (if installed on a single system), or onto the same system as any other IBM Tivoli Monitoring element in a distributed environment.
Response Time	Application Management Console	Install onto a stand-alone system, onto the same system as the IBM Tivoli Monitoring environment (if installed on a single system), or onto the same system as any other IBM Tivoli Monitoring element in a distributed environment. Do not install multiple copies of the same agent on the same computer. You can install only *one* Application Management Console (which is also the robotic script file depot) in the IBM Tivoli Monitoring environment. In systems with a high volume of transactions, install the Application Management Console on a separate computer. The Application Management Console is not a prerequisite to display agent data such as from Web Response Time in the Tivoli Enterprise Portal but you want to use this console because the Application Management Configuration Editor is required to configure all customizations such as transactions and filters. If the Application Management Console agent is not present, the default profiles are used when capturing data. The T1 (file transfer enablement) library is used to communicate with the Application Management Console, but is not required to upload data to the Tivoli Enterprise Monitoring Server or display the data in the Tivoli Enterprise Portal.
	Client Response Time Agent	
	Web Response Time Agent	
	Robotic Response Time Agent	

Table 7. Locating ITCAM for Transactions components (continued)

Component	Subcomponent	Possible location
Transaction Tracking See "Planning Transaction Tracking deployment" on page 45 for further information.	Aggregation agent, such as Transaction Collector	Install one Transaction Collector in each network segment or DMZ. For performance reasons, install each Transaction Collector on a separate computer outside the monitored domain. This task might require special configuration for some domains. See the configuration information for each domain in Chapter 9, "Configuring Transaction Tracking," on page 217.
	Transaction Reporter	Install the Transaction Reporter on the same computer as the Tivoli Enterprise Monitoring Server or on a separate computer in the same LAN segment.

Deploying monitors, monitoring agents, and Data Collector plug-ins

The monitors, monitoring agents, and Data Collector plug-ins that you require are dependent on what you want to monitor. Use Table 8 to help you find further information.

Table 8. Deploying monitors, monitoring agents, and Data Collector plug-ins

Component	Description	Installation	Further information
Internet Service Monitoring monitors	Each monitor tests 1 type of protocol or service.	Installed by the Internet Service Monitoring installer.	See Internet Service Monitoring reference in the *Administrator's Guide* for information about the Internet Service Monitoring monitors.
Response Time monitoring agents	Each monitoring agent monitors the client, server, or applications.	Installed by the Response Time installer.	See the following topics in the *Administrator's Guide* for further information: • Customizing Client Response Time • Customizing Robotic playbacks • Customizing Web Response Time
Transaction Tracking Data Collector plug-ins	Each Data Collector plug-in tracks transactions for a particular domain. You can combine several Data Collector plug-ins to track transactions across supported domains.	MQ Tracking and Data Collector for WebSphere Message Broker are installed by the Transaction Tracking installer. Install all other Data Collector plug-ins by using their separate, individual installers.	See Chapter 10, "Installing and configuring Transaction Tracking Data Collector plug-ins," on page 221 for information about the Transaction Tracking Data Collector plug-ins.

Prerequisites

Information about the required hardware and software for ITCAM for Transactions is available from, and regularly updated on the Prerequisite pages:

1. Go to ITCAM for Transactions on Documentation Central, and select the required ITCAM for Transactions version in the list.
2. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**.

Information to collect before you begin installation and configuration

Before you begin configuring the product, you must gather information about the communications protocol settings that the monitoring agent uses to communicate with the monitoring server. You have four choices: IP.UDP, IP.PIPE, IP.SPIPE, or SNA.

Tip: You can specify multiple communication methods. By using different methods, you can set up backup communication methods. If the method you have identified as Protocol 1 fails, the software uses Protocol 2, and so on.

Table 9. Communications protocol settings worksheet

Field	Description	Write the settings for your environment here
IP.UDP Settings		
Hostname or IP address	The host name or IP address for the hub monitoring server.	
Port number and Port Pools	The listening port for the hub monitoring server. The default port number is 1919.	
IP.PIPE Settings		
Hostname or IP address	The host name or IP address for the hub monitoring server.	
Port Number	The listening port for the monitoring server. The default port number is 1918.	
IP.SPIPE Settings		
Hostname or IP address	The host name or IP address for the hub monitoring server.	
Port number	The listening port for the hub monitoring server. The default port number is 3660.	
SNA Settings		
Network Name	The SNA network identifier for your location.	
LU Name	The LU name for the monitoring server. This LU name corresponds to the Local LU Alias in your SNA communications software.	

Table 9. Communications protocol settings worksheet (continued)

Field	Description	Write the settings for your environment here
LU 6.2 LOGMODE	The name of the LU6.2 LOGMODE. The default value is CANCTDCS.	
TP Name	The transaction program name for the monitoring server.	
Local LU Alias	The LU alias.	
General		
NAT Settings	The partition name and file of the NAT server.	

Planning to install Internet Service Monitoring

Internet Service Monitoring can be installed on AIX®, Linux, Solaris, UNIX, and Windows systems.

The installation uses an InstallShield Wizard to install and configure Internet Service Monitoring. Additional installation and configuration steps relate to the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and the Tivoli Enterprise Monitoring Server components of the IBM Tivoli Monitoring environment.

Hardware and software requirements

Internet Service Monitoring is available on a range of operating systems.

See the Supported operating systems for Internet Service Monitoring in the Prerequisites pages of the relevant ITCAM for Transactions Information Center for further information.

Installation prerequisites

Before installing Internet Service Monitoring, ensure that the target environment includes the list of required products, that they are configured, and that you have administrator access to their host computers.

Installation onto an IBM Tivoli Monitoring environment requires the following products. See the latest Prerequisite information in the ITCAM for Transactions Information Center for the required versions:

- Tivoli Enterprise Monitoring Server. Obtain the Tivoli Enterprise Monitoring Server IP address or Hostname before installation.
- Tivoli Enterprise Portal Server.

  If Tivoli Enterprise Portal Server uses Microsoft SQL Server 2000 as its database, you must manually create a user for that database with name `kis`, with roles `DB Creators` and `Bulk Insert Administrators`, and with access privileges `master = public, db_datareader, db_datawriter` and `teps = public, db_datareader, db_datawriter`.

  If Tivoli Enterprise Portal Server uses Microsoft SQL Server 2005 or Microsoft SQL Server 2008 as its database, add a **kis** schema to your Microsoft SQL Server Tivoli Enterprise Portal Server database before using the Internet Service Monitoring configuration tool in the Tivoli Enterprise Portal:

1. Open Microsoft SQL Server Management Studio.
2. Navigate to **Databases** > **teps** > **Security**.
3. Right click **Schemas** and select **New Schema**.
4. In the **General** tab, enter the following values:
   - **Schema name**: `kis`
   - **Schema owner**: `teps`
5. In the **Permissions** tab, enter the following values:
   - **Database**: `teps`
   - **Schema name**: `kis`
   - **Users or roles**: `[teps]`
   - **Explicit permissions for teps**: Grant permissions for Alter, Control, Delete, Execute, Insert, References, Select, Update
6. Restart the Tivoli Enterprise Portal client.

- Tivoli Enterprise Portal.
- Optional - Tivoli Data Warehouse and Summarization and Pruning Agent (equivalent version for IBM Tivoli Monitoring).

Installing Internet Service Monitoring purely as an event source for an IBM Tivoli Netcool/OMNIbus ObjectServer requires only IBM Tivoli Netcool/OMNIbus version 3.6 or higher. Before performing the installation, obtain the name of the target ObjectServer, for example `NCOMS`, the IP address or host name of the computer hosting that ObjectServer, and the port on which the ObjectServer listens.

Installation considerations

Before installing Internet Service Monitoring consider how you want to deploy the product, the size of the resources needed, the scalability of the Internet service monitors, and whether you require historical reporting.

Note: If you have previously installed Internet Service Monitoring, stop all Internet Service Monitoring monitors and the Databridge before installing any other IBM Tivoli Monitoring products. The locked files and processes from Internet Service Monitoring are not stopped automatically by the installer and will cause the Tivoli Enterprise Monitoring Agent installation to fail.

Planning deployment

If you want to use the graphical configuration, management, and reporting features of Internet Service Monitoring, you must install it into an IBM Tivoli Monitoring environment. This environment can be fully distributed. If you plan to use the product purely as an event source for IBM Tivoli Netcool/OMNIbus, IBM Tivoli Monitoring is not required.

Internet Service Monitoring can be installed onto a stand-alone system, onto the same system as the IBM Tivoli Monitoring environment (if installed on a single system), or onto the same system as any other IBM Tivoli Monitoring element in a distributed environment.

Internet Service Monitoring can be installed with any combination of the following:

- Tivoli Data Warehouse (for reporting of long term historical data)
- Summarization and Pruning Agent (for reporting of long term historical data with Tivoli Data Warehouse)
- Tivoli Enterprise Monitoring Server

- Tivoli Enterprise Portal Server
- Tivoli Enterprise Portal
- IBM Tivoli Netcool/OMNIbus (for use with IBM Tivoli Netcool/OMNIbus Event List)

Note: You can install Internet Service Monitoring at multiple sites for use with a single IBM Tivoli Monitoring environment.

Deployment scenarios:

Internet Service Monitoring can be deployed in a range of different network infrastructures.

Three possible deployment scenarios for Internet Service Monitoring on service provider network infrastructures are described. Also provided are guidelines on the size and scale of the deployment. These factors are important because service monitoring activities use network, disk, and database resources.

Deployment within an ISP infrastructure:

Deploying Internet Service Monitoring outside the Internet Service Provider (ISP) firewall on the De-Militarized Zone (DMZ) LAN, enables monitoring of Internet services inside and outside the firewall.

Figure 6 on page 36 shows a sample Internet Service Monitoring deployment in a simple ISP infrastructure.

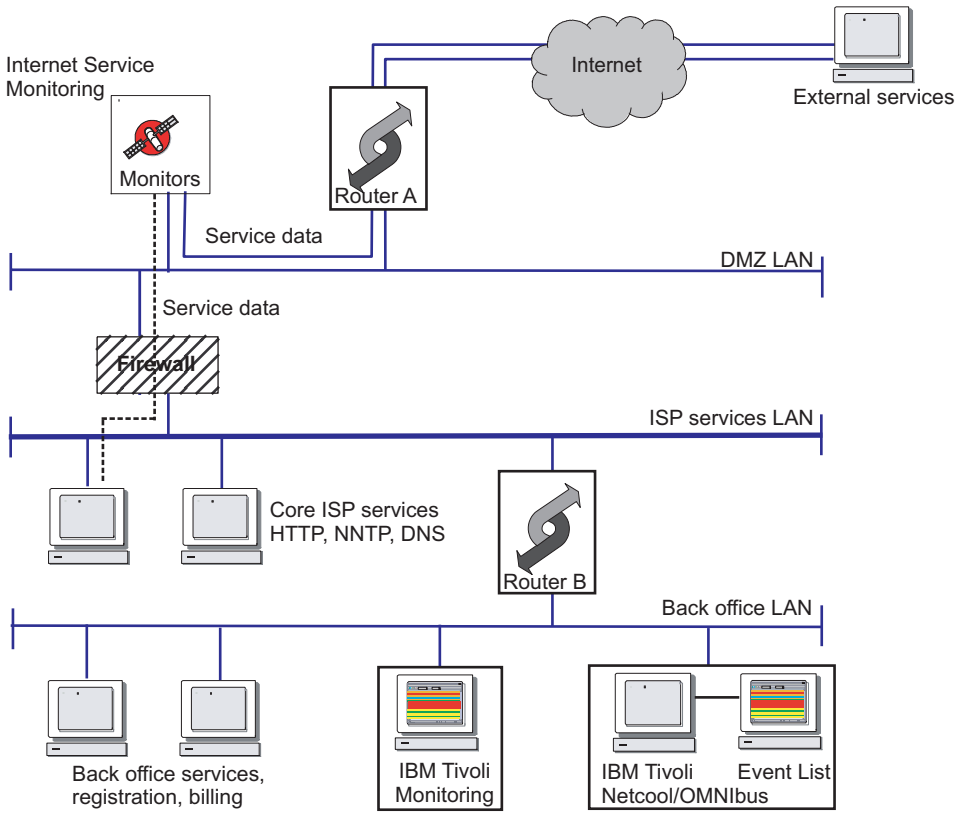


Figure 6. Deployment within an ISP infrastructure

The figure shows Internet Service Monitoring installed outside the ISP firewall on the De-Militarized Zone (DMZ) LAN. This installation enables the monitors to emulate one of the customers for the ISP. Monitors installed on the DMZ LAN monitor Internet services inside or outside the firewall. For example, the HTTP monitor might monitor web pages served from the ISP Services LAN behind the firewall, and other web pages located elsewhere on the Internet through Router A which connects the DMZ LAN to the Internet.

Internet Service Monitoring sends results through the firewall and Router B to IBM Tivoli Monitoring, which is connected to the Back Office LAN. From there, operators manage all Internet services and infrastructure in workspaces. Internet Service Monitoring also sends alerts through the firewall and Router B to IBM Tivoli Netcool/OMNIbus. IBM Tivoli Netcool/OMNIbus is connected to the Back Office LAN. From there, operators view the Event List and manage any problems.

Deployment within an MNS provider infrastructure:

Deploying Internet Service Monitoring outside the Managed Network Service (MNS) firewall on the De-Militarized Zone (DMZ) LAN, enables monitoring of Internet services inside the firewall.

Figure 7 on page 37 shows a sample Internet Service Monitoring deployment in a simple MNS provider infrastructure.

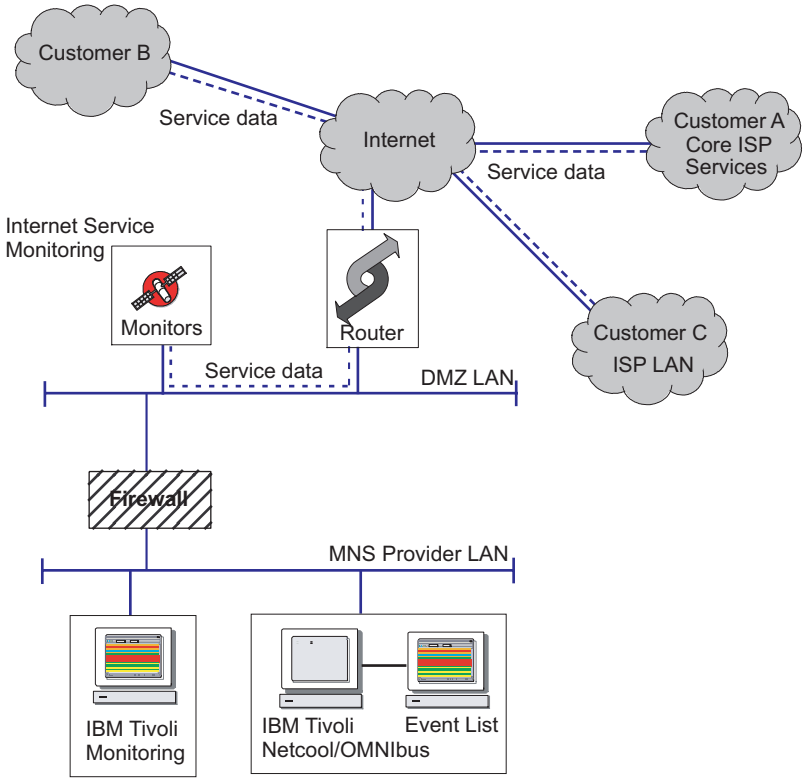


Figure 7. Deployment within an MNS provider infrastructure

The figure shows Internet Service Monitoring installed outside the firewall for the MNS provider on the DMZ LAN. From outside the firewall, the monitors connect to customer networks to monitor their services.

Internet Service Monitoring sends results through the firewall to IBM Tivoli Monitoring. From there, operators view the results in workspaces.

Internet Service Monitoring also sends alerts through the firewall to IBM Tivoli Netcool/OMNIbus. From there, operators view the event list and manage any problems.

Deployment within a distributed ISP infrastructure:

Many Internet Service Providers (ISPs) and large corporate intranet service providers use a distributed infrastructure to reduce WAN requirements and increase server performance. In this situation Internet Service Monitoring is installed within each Point Of Presence (POP).

Service providers can deploy Internet Service Monitoring to monitor the distributed services that they supply to their corporate customers.

Figure 8 on page 38 shows a sample of an infrastructure with multiple POPs.

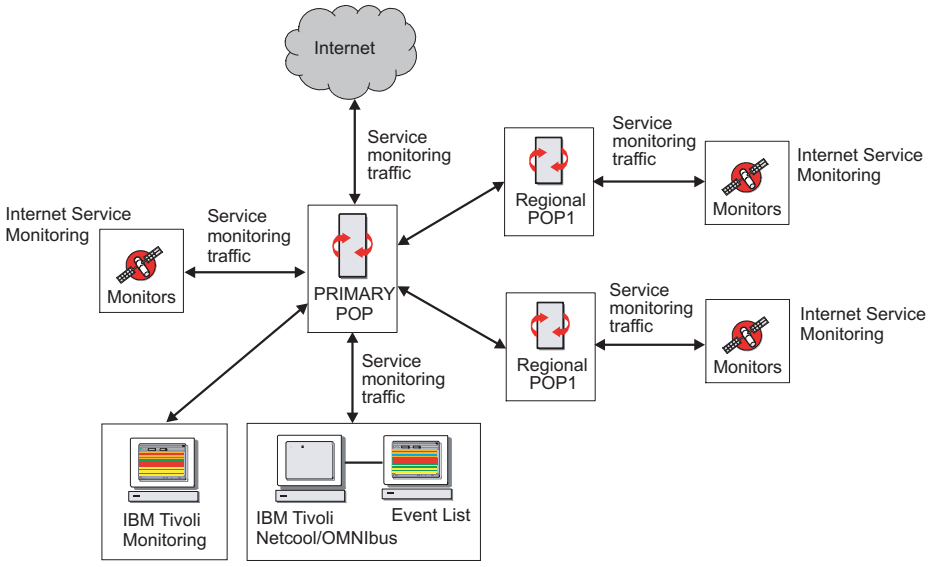


Figure 8. Deployment within a distributed ISP infrastructure

The figure shows Internet Service Monitoring installed within each POP. Service monitor traffic, such as polls and responses, between the POPs are sent to IBM Tivoli Monitoring. From there, operators view the results in workspaces. Service monitor traffic is also sent to IBM Tivoli Netcool/OMNIbus. From there, operators view the event list and manage any problems.

Fault-tolerant deployment:

The simplest configuration providing fault-tolerant operation is to create two separate Internet Service Monitoring installations, and distribute profiles from one installation to the other.

Both installations then perform identical monitoring tasks, so the monitoring operations are duplicated. Performance data (events and test results) and datalog files are also duplicated. This type of installation is suitable for disaster recovery requirements.

Note: The Internet service monitors and the ObjectServer module of the Databridge implement the store and forward features of the IBM Tivoli Netcool/OMNIbus Probe API. These features prevent the loss of performance data in the case of network connectivity loss or component failure.

Sizing guidelines

Service monitoring activities use network, disk, and database resources. For this reason, guidelines are provided to assist you with sizing and scaling of the Internet Service Monitoring deployment.

The formulas presented here are intended as a guide only. The exact requirements might vary according to operating environment and service monitoring application.

Network bandwidth use:

Monitoring activities and forwarding of events to a IBM Tivoli Netcool/OMNIbus ObjectServer use network bandwidth. The bandwidth that is required depends on the size of the data, number of monitoring requests, and frequency of the requests. Guidelines are provided to assist with estimating the required network bandwidth.

Monitor traffic

Note: The values described for bandwidth usage represent an upper limit, real values are likely to be lower.

HTTP monitor tests commonly comprise a large proportion of the incoming traffic to Internet Service Monitoring. A `GETALL` request to download a web page and its components, assuming a total size of 20 Kb size, requested every 10 minutes corresponds to network bandwidth use of 266.67 bps. 10,000 requests every 10 minutes, which is the limit of the HTTP monitor performance capability, corresponds to bandwidth use of 2.67 Mbps.

Tip: The more commonly used HTTP `GET` and `HEAD` requests use less bandwidth than `GETALL` requests.

ObjectServer events

The size of each HTTP event sent from the Databridge to a IBM Tivoli Netcool/OMNIbus ObjectServer, is approximately 2Kb. Running one HTTP monitor test every 10 minutes corresponds to network bandwidth use of 26.67 bps, outgoing from Internet Service Monitoring. 10,000 tests every 10 minutes corresponds to bandwidth use of 0.267 Mbps.

Disk space requirements:

Datalogs are created by the Databridge Datalog module. These datalogs can occupy a large amount of disk space.

Note: Typically the Datalog module is not required; it is disabled by default. To calculate the amount of disk space required by datalog files, use the following formula:

```
(600 / poll_interval) × 15 KB per profile element per day
```

For example, a single HTTP profile element polling a web page every 5 minutes for one year would require 10.95 MB of disk space:

```
(600 / (5 × 60)) × 15 × 365 = 10.95 MB
```

Monitor scalability and performance

The scalability and performance of Internet service monitors directly affects the volume of services that you can monitor.

Scalability and performance depend on a number of factors:

- Access speed of the disk system used by the monitors
- Response times of the monitored services
- Number of monitor threads running concurrently
- Time between tests
- Number of profiles

- Monitor poll interval

The monitoring environment determines the disk speeds and service response times. Monitor properties control concurrent threading and the time between tests. You control the number of profiles and the monitor polling intervals when you configure the monitors through the Internet Service Monitoring user interface in the Tivoli Enterprise Portal.

Monitor property settings:

Internet service monitor properties include settings for the maximum number of threads, the interval at which the monitor spawns threads, and the maximum size of the event queue for the monitor. Ensure that these settings are optimized.

Table 10 shows the optimal settings for the `MaxCCA`, `Pause`, and `QSize` properties.

Table 10. Optimal settings for Internet service monitor properties

Property	Description	Best setting
`MaxCCA` *n*	Sets the maximum number of threads running concurrently. The ICMP monitor does not provide this property because it is single-threaded.	`50` for high-volume polling (more than 500 profile elements) `10` for low-volume polling
`Pause` *n*	Sets the interval (in milliseconds) at which the monitor spawns threads. Setting it to a higher value, such as 100 or more, causes the monitor to spawn threads more slowly.	`50` for high-volume polling (more than 500 profile elements) `100` for low-volume polling
`QSize` *n*	Set the maximum size of the monitors event queue on disk.	`10` MB for most situations. In larger environments, increase this value for each monitor by 1 MB per 1000 elements.

Poll intervals:

The number of poll intervals affects the response times. A guideline is provided to help with choosing the appropriate interval.

You define the poll interval when you create profile elements. Use the following formula as a guide:

```
minimum poll interval = number of profile elements × average response time / MaxCCA
```

The number of profile elements configured for a monitor determines the total number of tests performed by the monitor. The average response time varies according to the monitoring environment, so when selecting the poll interval, choose a value appropriate to the application environment. In the worst case, if the application fails to respond then the average response time is the timeout value set for the monitored element.

Note: If you use history collection, use the same poll intervals as the history collection intervals.

Response times in LAN monitoring environments

In LAN monitoring environments, server tests run over high-speed networks. Therefore the response times of services are low, typically less than one second.

The poll interval is also affected by data logging functions that involve a higher number of disk access operations. If you use monitors as data sources for only IBM Tivoli Netcool/OMNIbus and IBM Tivoli Monitoring, and you do not use the datalogs as storage for simple reporting or archiving results, then monitor data logging is not necessary and you can run the monitors without data logging. This condition permits shorter polling intervals of up to one half of the value indicated by the polling interval formula.

Response times in remote monitoring environments

In remote monitoring environments, where service tests run over a WAN or an Internet connection, the response times are less predictable, and it is the network response time, instead of monitor performance, that limits the poll interval. Use polling intervals that allow enough time for responses to be received before the next test starts.

Historical reporting

The Internet service monitoring workspaces in Tivoli Enterprise Portal show the most recent test results. By default, these results are the results of the last hour. You can configure IBM Tivoli Monitoring to store the test results thus providing access to historical data for short-term or long-term reporting in the history workspaces.

Short-term data refers to data that is available for up to 24 hours. Long-term data refers to data that is available for an indefinite time.

Short-term data reporting:

You can save the most recent test results for up to 24 hours to enable short-term historical reporting. Short-term historical data is collected and stored in binary files.

Use the History data collection feature in Tivoli Enterprise Portal to configure the collection of historical data. See the Administrator's Guide for information about configuring history collection.

Long-term reporting:

You can save test results indefinitely to enable long-term historical reporting. Long-term historical data is stored in the Tivoli Data Warehouse.

Use the History data collection feature in Tivoli Enterprise Portal to configure the collection of historical data. See the Administrator's Guide for a description of the configuration fields.

Installation of support files

In addition to installing Internet Service Monitoring, you must install support files for the IBM Tivoli Monitoring components that are used by Internet Service Monitoring.

Support files add support information for the predefined workspaces and situations to the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server components of IBM Tivoli Monitoring. There is a support file for each of these components.

Support files must be installed on the same computer as the components they are supporting. For example, Tivoli Enterprise Monitoring Server Install support on the same computer as Tivoli Enterprise Monitoring Server.

The support files are included on the product CD or downloaded from the IBM Passport Advantage® website http://www.ibm.com/software/howtobuy/passportadvantage/ as part of the product. You install the files by using the Internet Service Monitoring InstallShield Wizard.

If you intend to use Internet Service Monitoring purely as an event source for IBM Tivoli Netcool/OMNIbus, installing support files is not necessary.

Planning to install Response Time

Administrators can install Response Time monitoring agents and related software.

Prerequisites

Information about the required hardware and software is available and regularly updated on the Prerequisite web pages in the ITCAM for Transactions Information Center.

Important: If you are installing agents by using Tivoli Configuration Manager and `SPD` files, only Tivoli Configuration Manager version 4.1.1 is supported for this release.

Considerations for installing Response Time

Before installing Response Time

Before installing Response Time monitoring agents, ensure that the following conditions are satisfied:

- You must have **Administrator** privileges to install Response Time monitoring agents.
- You are installing only **one** Application Management Console agent (which is also the robotic script file depot) in the IBM Tivoli Monitoring environment.
- You are not installing multiple copies of the same agent on the same computer.
- The Application Management Console must be on its own server-class box if monitoring more than 50 agents or if using Response Time to monitor a high volume web site generating thousands of transactions an hour. In test environments, it can be on the same computer as the Tivoli Enterprise Monitoring Server and the Tivoli Enterprise Portal.
- The Tivoli Enterprise Portal has a 26 character limitation for the host and application names in the Navigator. When you have a combination of names that are longer than 26 characters, the application name is truncated. For

example, if you have application names such as *Websphere Plants* and *Websphere Petstore* on a computer with a long host name, the names might be displayed in the Navigator as `WebspherePl` and `WebspherePe`, whereas if the host name are only four characters shorter they are displayed as `WebspherePlants` and `WebspherePetsto`. To avoid truncating the host and application names in the Navigator, install the Application Management Console on a computer with a short host name.

- If you have a previously installed Response Time Tracking, version 6.1, you must uninstall it from your environment and verify that the following .DLL files are deleted before installing Robotic Response Time or Client Response Time.
  - `c:\windows\system32\libarm4.dll`
  - `c:\windows\system32\libarm32.dll`
  - `c:\windows\system32\armjni4.dll`
  - `c:\windows\system32\armjni.dll`
  - `c:\windows\system32\armcli.dll`
  - `c:\windows\system32\libarm4net.dll`

  where `c:\windows\system32` is the Windows `SYSTEM32` directory.
- If you want to record or playback IBM Rational Robot scripts, you must install IBM Rational Robot before installing Robotic Response Time.
- Before installing the Web Response Time agent, you must install the following prerequisites:
  - Windows Network Monitor or Windows packet capture library (see "Installing Windows Network Monitor or Windows packet capture library" on page 126).
  - Microsoft Visual C++ 2008 SP1 Redistributable Package (x64) (see the Microsoft web site download center for more information)

  If you do not install these packages before you install Web Response Time, the **`kfcmserver`** process does not start, and response time data does not display in the Tivoli Enterprise Portal.
- If you want to perform a Remote Deployment, go to Chapter 11, "Working remotely," on page 309.
- When installing Client Response Time (t4) or Robotic Response Time (t6), you must manually stop any applications that use ARM before starting the installation.

Planning to install Transaction Tracking

Before you install Transaction Tracking, consider installation requirements and prerequisites.

Hardware and software requirements

Transaction Tracking is available on a range of operating systems.

Supported operating systems

Transaction Tracking is supported on the operating systems shown in the Transactions Prerequisite Web pages.

Software requirements

- Time synchronization software

If you are using multiple Transaction Collectors, use UTC and Network Time Protocol (NTP) or similar time synchronization software to synchronize the clocks for the Transaction Collectors. This method enables Transaction Tracking to provide an accurate timestamp for transactions and accurately calculate transactions times.

For ITCAM for Transactions V7.2.0.2 and later, if you are using a single Transaction Collector, time synchronization software is not required. The Transaction Collector calculates the time differences between data collectors and adjusts the timestamps of their events automatically. However, if the time variation between the data collectors and the Transaction Collector is more than one week or less than 1 minute, the timestamps are not adjusted.

- Java runtime Environment, IBM JRE version 1.4 or 1.5

Installation prerequisites

Before installing Transaction Tracking, ensure that the target environment includes the required products, that they are configured, and that you have administrator access to their host computers.

You must have the following products installed in your IBM Tivoli Monitoring environment before you can install Transaction Tracking. See the latest Prerequisite information in the ITCAM for Transactions Information Center for the required versions:

- Tivoli Enterprise Monitoring Server.

  Obtain the Tivoli Enterprise Monitoring Server IP address or host name, communication protocol, and listening port before installation. If your system uses a firewall or network address translation system, look up the IBM Tivoli Monitoring values in these systems.
- Tivoli Enterprise Portal Server with the topology evaluator installed.
- Tivoli Enterprise Portal.
- Tivoli Data Warehouse and Summarization and Pruning Agent (equivalent version for IBM Tivoli Monitoring).

  If you do not install the Tivoli Data Warehouse you will be able to view only a limited period of historical data, which is 24 hours by default.
- Application Management Console agent.

Tip: If your system uses a firewall or network address translation system, look up the IBM Tivoli Monitoring values in these systems.

Note: If you have previously installed Internet Service Monitoring, stop all Internet Service Monitoring monitors and the Databridge before installing any other IBM Tivoli Monitoring products. The locked files and processes from Internet Service Monitoring are not stopped automatically by the installer and will cause the Tivoli Enterprise Monitoring Agent installation to fail.

Required access privileges for installing

You must have administrator access to the host computers of the IBM Tivoli Monitoring environment to which you intend to install Transaction Tracking.

On UNIX or Linux systems, you might prefer a non-root installation.

Note: If you have installed other Tivoli products as root, check the permissions of `/var/tmp/plugin_tmp` to ensure that the non-root user you want to use to install IBM Tivoli Composite Application Manager for Transactions has permission to do so.

You must also know the IP addresses or host names of the IBM Tivoli Monitoring components.

Planning Transaction Tracking deployment

To install Transaction Tracking, install an agent and support for several components as well as support files which update IBM Tivoli Monitoring components.

Installation guidelines

Read the guidelines for Transaction Tracking. See "Planning ITCAM for Transactions deployment and installation" on page 25.

When installing Transaction Tracking into your IBM Tivoli Monitoring environment, consider the following guidelines:

- Install one or more Transaction Reporters.
- Install one or more Aggregation agents:
  - Install one Transaction Collector in each network segment or DMZ.
  - For performance reasons, install each Transaction Collector on a computer separate to the monitored domain.
  - If using remote Transaction Collectors with ARM-based data collectors; such as IBM HTTP Server, Rational Performance Tester, WASTT, or ARM-instrumented WebSphere Application Server; special consideration is required.

    Profiles configured in the Application Management Configuration Editor are distributed to the Transaction Collector and from there to the ARM data collectors. To distribute the profiles automatically, the Transaction Collector must be located on the same computer as the ARM data collector. Consider installing a *configuration-only* Transaction Collector on the same computer as the ARM data collectors, and use that Transaction Collector only to distribute Application Management Configuration Editor profiles. Continue to send transaction events to the remote Transaction Collector.

    Alternatively, distribute the profiles to the ARM data collectors manually. See Manually synchronizing Transaction Tracking profiles with the Application Management Console agent in the *Administrator's Guide* for further information.
  - Install other Aggregation agents as required.
  - If you install more than one Aggregation agent, such as Transaction Collector and Web Response Time, and the agents provide data to the same Transaction Reporter, ensure that the clocks of the computers on which they are installed are synchronized.

- Install Transaction Tracking support files in Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal desktop client.
- Install Application Management Console (t3 agent) if you want to configure ARM in your deployment. See Chapter 6, “Installing Response Time,” on page 101 for further information.
- Transaction Tracking uses TCP port **5455** for communication between the data collectors (such as the ARM data collector) and the Transaction Collector. If you are using firewalls, allow Transaction Tracking to use this port.
- Use UTC and Network Time Protocol (NTP) or similar time synchronization software to synchronize the clocks for the Transaction Reporter, and Aggregation agents such as Transaction Collector and Web Response Time.

Downloading Transaction Tracking software

Install Transaction Tracking from the product CD or download it from the IBM Passport Advantage® website.

Procedure

To download the product from the IBM Passport Advantage® website:

1. Go to http://www.ibm.com/software/howtobuy/passportadvantage/.
2. Click **Customer sign in** and sign in using your IBM ID and password.
3. Select **Software download & media access**.
4. Click **Download finder**, then select **Find by product description** and type in **Tivoli**.
5. Expand the Tivoli Software, navigate to IBM Tivoli Composite Application Manager for Transactions and select the images that you want to download.

Chapter 3. Installing ITCAM for Transactions by using the ITCAM for Transactions installer

Install ITCAM for Transactions by using the ITCAM for Transactions installer or install each component separately.

By using the ITCAM for Transactions installer, also known as the *solution installer*, you can install all components of ITCAM for Transactions simultaneously, including:

- Internet Service Monitoring
- Response Time
- Transaction Tracking

You can also install the base IBM Tivoli Monitoring product, including an IBM DB2 database server (a prerequisite for the Tivoli Enterprise Portal Server component of IBM Tivoli Monitoring, if you do not already have a supported database server application installed).

The ITCAM for Transactions installer has two types of installation:

- Use *Basic Installation* to install a complete IBM Tivoli Monitoring environment including DB2 and all components of ITCAM for Transactions for an entirely new installation.
- Use *Custom Installation* to install only the servers, agents, and application support that you require separately, as part of an existing IBM Tivoli Monitoring environment.

Avoiding a port conflict with DB2: If you install DB2 Workgroup Server Edition version 9.5, manually stop the Monitoring Agent for DB2. This agent is installed automatically with DB2, and uses port 1920. When you later launch the ITCAM for Transactions installer to install IBM Tivoli Monitoring and ITCAM for Transactions agents, the ITCAM for Transactions installer checks that port 1920 is available before installing the Tivoli Enterprise Monitoring Server. If the Monitoring Agent for DB2 is active, you might receive the message **Port 1920 is occupied**.

Prerequisites

Information about the supported browsers for the ITCAM for Transactions installer is available from the Prerequisite pages:

1. Go to ITCAM for Transactions on Documentation Central, and select the required ITCAM for Transactions version in the list.
2. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**.
3. Select any component in the **Compatible software** list.

Note: The ITCAM for Transactions installer is not supported on Linux on Z series systems.

Product codes

Table 11 lists the product codes of the individual components that make up ITCAM for Transactions. You might need the product code if you want to install individual components, or to perform operations from the command line.

Table 11. ITCAM for Transactions product codes

Component	Subcomponent	Product code
Internet Service Monitoring		`is`
Response Time	Application Management Console	`t3`
	Client Response Time	`t4`
	Web Response Time	`t5`
	Robotic Response Time	`t6`
Transaction Tracking	Transaction Collector	`tu`
	Transaction Reporter	`to`
	MQ Tracking	`th`
	Data Collector for WebSphere Message Broker	`k3`

Getting started

Regardless of whether you are doing a basic or custom installation, the initial steps are the same.

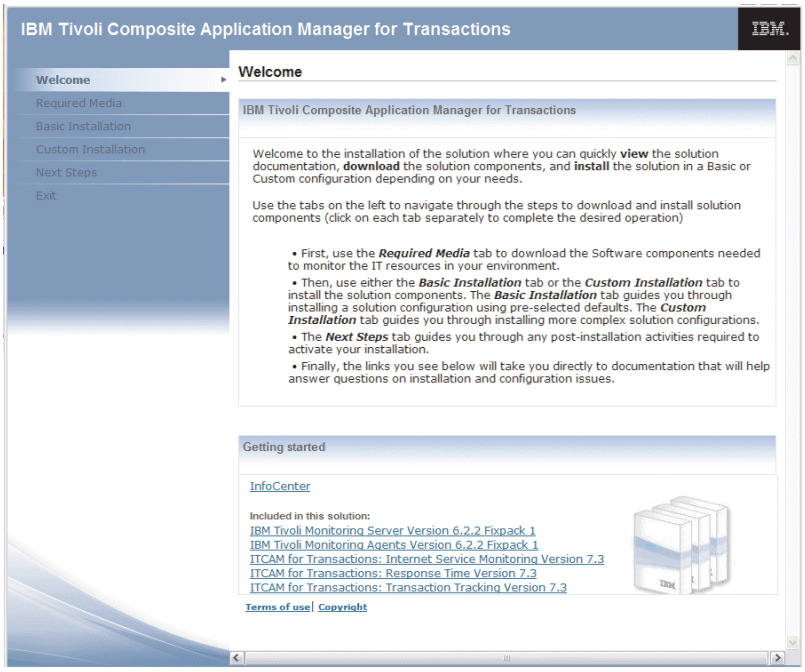
The ITCAM for Transactions installer is included on the product DVD or can be downloaded from IBM Passport Advantage®. For information about obtaining the latest version, see the Download information about the ITCAM for Transactions Information Center.

To start installing:

1. Log on by using a user ID with Administrator authority.
2. Obtain the installation software by downloading it to a local destination directory or inserting the product DVD.

   Note: Do not use a network drive or shared network folder as the installation destination for the ITCAM for Transactions installer.
3. Extract the compressed file to a folder on your local drive.
4. Start the installation wizard by running `launchpad.bat` on Windows systems, or `launchpad.sh` on Linux and UNIX systems. The installation wizard is loaded. Click **OK** to continue.

5. The **Welcome** page is displayed.



The navigation pane on the left shows the steps to follow to install ITCAM for Transactions. The pane on the right provides further information.

6. Select **Required Media** in the navigation pane. The **Required Media** page lists the ITCAM for Transactions components and their eAssembly Image Part Numbers. If you are not installing from a DVD, use this information to download the components that you want to install from IBM Passport Advantage. Also see the Download information about the ITCAM for Transactions Information Center for further information.

   If you have already downloaded the ITCAM for Transactions installation images, the ITCAM for Transactions installer automatically discovers them. See "Installing new components by using ITCAM for Transactions installer basic installation" for further information.

Installing new components by using ITCAM for Transactions installer basic installation

Use the Basic Installation option in the ITCAM for Transactions installer to install new versions of the ITCAM for Transactions components, the IBM Tivoli Monitoring framework, and a DB2 database server in an evaluation or small production environment. After the initial setup, all components are installed silently to the same computer by using default configuration settings.

Before you begin

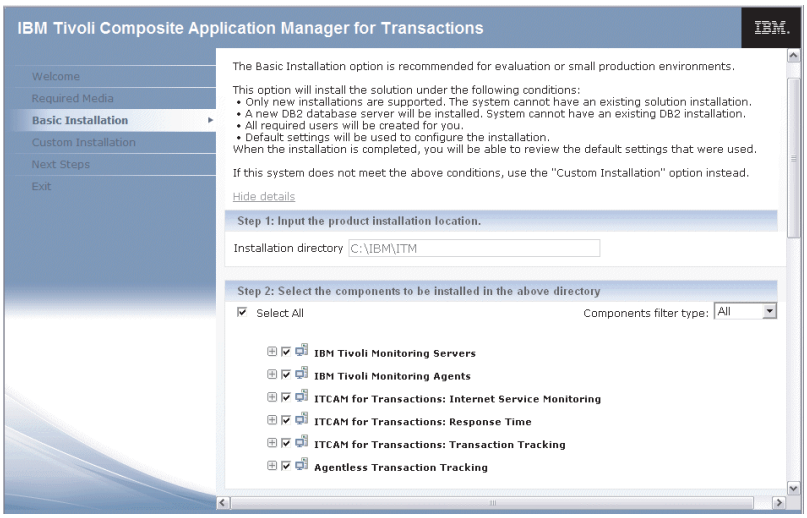
Complete the preparatory steps described in "Getting started" on page 48.

About this task

To install new versions of ITCAM for Transactions and the IBM Tivoli Monitoring framework:

Procedure

1. Select **Basic Installation** in the navigation pane:



2. In Step 1 in the **Installation directory** field, the default installation location is displayed. Click **Change** and, if required, browse to a new location. If a product that uses the IBM Tivoli Monitoring framework is already installed, the installation directory is detected automatically and cannot be updated.
3. In Step 2, select the components to install. You can expand the selection nodes to display the list of individual components to be installed. Move the cursor over each component to display its product code.

   If the prerequisites for a component are not met, it is not available. Click **i** for an explanation.

   If you select **IBM Tivoli Monitoring Agents Support file**, or the support file for any agent, the required **IBM Tivoli Enterprise Monitoring Server**, **IBM Tivoli Enterprise Portal Server**, and **IBM Tivoli Enterprise Portal desktop client** are also selected automatically.

   If you are using an existing database server (IBM DB2 or other supported application), expand the **IBM Tivoli Monitoring Servers** node. Then expand the IBM Tivoli Enterprise Portal Server node, and enter the database details.

As an alternative to selecting components from each major component group, you can also select the **Agentless Transaction Tracking** option. This option installs the ITCAM for Transactions Web Response Time agent (from the Response Time agent group) and the Transaction Reporter (from the Transaction Tracking agent group). You can also manually select these same agents directly from each component group.

The installer calculates the disk usage and file system permissions.

4. In Step 3, select the installation image location. To discover all available images automatically:
   a. Select **Install from local image top level directory** and click **Search**.
   b. In the IBM Tivoli Composite Application Manager for Transactions dialog box, enter a directory for the ITCAM for Transactions installer to search, or click **Browse** and navigate to it. Limiting the search to a specific directory reduces the time taken to conduct the search.

      Note: If you are using Mozilla Firefox as your default web browser, you might encounter problems when your search scope is too large. This situation causes the installation program to fail. Use a narrow search scope to locate the images, or upgrade your installation of Firefox to the latest version.

c. Click **OK**. The number of images found is displayed in a results dialog box. The download location for each image is listed in Step 3 under the corresponding product.

5. In Step 4, click the link to each license agreement, read it, and click **Accept**.
6. Click **Install**.
7. The ITCAM for Transactions installer checks for prerequisite software. If a prerequisite is missing, a **Validation** panel is displayed. You do not need to close the ITCAM for Transactions installer. Instead, correct the problem and click **Retry**. The installation status is shown in the **Installation** pane.
8. When the installation is complete, the **Installation Summary** displays the results for each component. If required, click **Save** to save a detailed installation report as HTML.

Installing components by using ITCAM for Transactions installer custom installation

Use Custom Installation in the ITCAM for Transactions installer to install IBM Tivoli Monitoring, the agents, and support for those agents separately.

Before you begin

Complete the preparatory steps described in "Getting started" on page 48.

Note: For some non-English locales, the directory paths might not be displayed with correct path delimiters. This problem is a known limitation of the installation program but it does not affect functionality.

About this task

To selectively install IBM Tivoli Monitoring agents, and support for those agents:

Procedure

1. Select **Custom Installation** in the navigation pane:
   - Select **Server Installation** to install IBM Tivoli Monitoring server components by using the IBM Tivoli Monitoring common installer wizard.
   - Select **Agent Installation** to install IBM Tivoli Monitoring agent components including ITCAM for Transactions by using the IBM Tivoli Monitoring common installer wizard.
   - Select **Application Support Installation** to install agent application support for the Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, or Tivoli Enterprise Portal desktop client by using the IBM Tivoli Monitoring common installer wizard.

     The option to install support files is available only if the application to which it relates is already installed.
2. Click **Install**. The IBM Tivoli Monitoring common installer wizard is launched. See the installation information for each component for further information:
   - Chapter 4, "Installing Internet Service Monitoring," on page 55
   - Chapter 6, "Installing Response Time," on page 101
   - Chapter 8, "Installing Transaction Tracking," on page 187

Configuring components by using the ITCAM for Transactions installer

After you have installed the ITCAM for Transactions components and IBM Tivoli Monitoring framework by using either basic or custom installation, you can use the ITCAM for Transactions installer to configure those components.

Before you begin

Complete the preparatory steps described in "Getting started" on page 48.

About this task

To configure ITCAM for Transactions or IBM Tivoli Monitoring framework components:

Procedure

1. Select **Next Steps** in the navigation pane.
2. In the **Configure and activate agents** section, expand the component and click **Configure Agent** for the agent that you want to configure.

   The configuration dialog box for the selected agent is displayed. See the configuration information for each component for further information:
   - Chapter 5, "Configuring Internet Service Monitoring," on page 83
   - Configuring Response Time and related software
   - Chapter 9, "Configuring Transaction Tracking," on page 217
3. Update the required values on the various tabs and click **OK**.

   TEMS configuration not included: When you use this procedure to configure the agents, this process does not include the configuration of the Tivoli Enterprise Monitoring Server settings. To configure the Tivoli Enterprise Monitoring Server settings in addition to the agents, launch the configuration from the Manage Tivoli Enterprise Monitoring Services console. Use the **Reconfigure** option for each agent as needed.
4. When you have finished configuring the agents, start the Tivoli Enterprise Portal. In the **Start the Tivoli Enterprise Portal** section:
   a. Click the link to start the Tivoli Enterprise Portal in a browser.
   b. Click **Start Tivoli Enterprise Portal** to start the Tivoli Enterprise Portal desktop client.

What to do next

Select **Exit** in the navigation pane to close the ITCAM for Transactions installer. If you want to save your installation preferences, click **Save**.

Chapter 4. Installing Internet Service Monitoring

Internet Service Monitoring can be installed on AIX, Linux, Solaris, UNIX, and Windows systems.

The installation uses an InstallShield Wizard to install and configure Internet Service Monitoring. Additional installation and configuration steps relate to the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and the Tivoli Enterprise Monitoring Server components of the IBM Tivoli Monitoring environment.

Installing Internet Service Monitoring on Windows systems

You can install IBM Tivoli Composite Application Manager for Transactions to a system that also has other IBM Tivoli Monitoring components installed, or you can install IBM Tivoli Composite Application Manager for Transactions to a separate system.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

To install the IBM Tivoli Composite Application Manager for Transactions agent to a separate, Windows based, system:

1. Log on as a user with administrative privileges.
2. Insert the product DVD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage/.
3. Launch the installation wizard by double-clicking the `setup.exe` file. The InstallShield Wizard starts.

   Tip: On Windows Server 2008 systems, if instead of the installer you see the following message, right-click the `setup.exe` file in the file explorer and select **Run as Administrator**.

   ```
   Your logon ID must have Administrator rights to install IBM Tivoli Composite
    Application Manager for Transactions
   ```
4. On the **Welcome** window, click **Next**.
5. If no IBM Tivoli Monitoring components are installed on this computer the **Prerequisites** window is displayed. Read the information and click **Next**.
6. On the **Install Prerequisites** window, options to ensure that you have the correct version of IBM GSKit or IBM Java are selected. Click **Next**. The required software is installed automatically.
7. On the **Software License Agreement** window, read the agreement and click **Accept**.

8. If you install to a computer that does not have other IBM Tivoli Monitoring components installed, the **Choose Destination Location** window with the default installation location is displayed. Change the location if required and click **Next**.
9. On the **User Data Encryption Key** window, enter your own unique encryption key and click **Next** then click **OK** in the summary window.

   Note: You are only required to supply an encryption key if the IBM GSKit is not already installed on the computer. Use the same key across the enterprise.
10. On the **Select Features** window, select **Tivoli Enterprise Monitoring Agents** and click **Next**. This selects the **Tivoli Enterprise Monitoring Agent Framework** and the **IBM Tivoli Composite Application Manager for Transactions agent**.

    Note: Tivoli Enterprise Monitoring Agents may already be selected if the framework or any agents are already installed. You must expand this feature and also select **IBM Tivoli Composite Application Manager for Transactions agent**.
11. On the **Agent Deployment** window, select **Internet Service Monitoring** if you want to deploy to a remote location by using the Tivoli Enterprise Monitoring Server and click **Next**. If you are installing locally, do not select any agents.

    See Chapter 11, "Working remotely," on page 309 for further information.
12. On the **Start Copying Files** window, review the settings and if correct, click **Next**. The files are then copied. A **Setup Status** window informs you about progress.
13. On the **Setup Type** window, select all configuration options and click **Next**.

    You can delay some configuration until after installation if required. The following steps assume that all configuration options are selected.
14. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:
    a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.
    b. Select **Protocol 1** and select a protocol from the list. Several types of protocols are available: IP.UDP (uses unsecured UDP communications), IP.PIPE (uses unsecured TCP communications), IP.SPIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
    c. If additional protocols are required, select **Protocol 2** and select an second protocol from the list.
    d. Do not select **Optional Secondary TEMS Connection**. You can set up the failover support for the component later. See the *IBM Tivoli Monitoring User's Guide* for further information.
    e. Click **OK**.
15. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**.

    These settings define communications between the agent and the Tivoli Enterprise Monitoring Server. The host name or IP address of the local computer are displayed unless the Tivoli Enterprise Monitoring Server has already been specified. Ensure that you enter the host name or IP address of the Tivoli Enterprise Monitoring Server in the **Hostname or IP address** field if it is installed on another computer. The default port number for the previously selected protocol is also displayed (IP.PIPE is 1918, IS.SPIPE is 3660).

16. On the **Internet Service Monitoring Configuration** window, if using IBM Tivoli Netcool/OMNIbus select **YES** and enter the host name or IP address of the IBM Tivoli Netcool/OMNIbus ObjectServer and its port number.

    Tip: If you are using IBM Tivoli Netcool/OMNIbus but want to configure the connection later, select **NO**. Configure the ObjectServer connection later using the Manage Tivoli Enterprise Monitoring Services.
17. Click **OK**. The system configures and starts IBM Tivoli Composite Application Manager for Transactions.
18. Read the README file and click **Finish**. The **Manage Tivoli Enterprise Monitoring Services** window is displayed.
19. Verify the installation and configuration by checking the **Status** column for IBM Tivoli Composite Application Manager for Transactions. The **Status** column should show **Started**.

Results

Installation is complete.

What to do next

You can now install the support files and configure the product components.

Tip: Reconfigure connection to the Tivoli Enterprise Monitoring Server after installation using the Manage Tivoli Enterprise Monitoring Services.

Installing Tivoli Enterprise Monitoring Server support on Windows systems

Install Tivoli Enterprise Monitoring Server (TEMS) support to the system on which Tivoli Enterprise Monitoring Server is installed. Installation of Tivoli Enterprise Monitoring Server support on Windows also adds the application support.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

1. Log on as a user with administrative privileges to the system running Tivoli Enterprise Monitoring Server.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage/.
3. Launch the installation wizard by double-clicking the `setup.exe` file. The InstallShield Wizard starts.
4. Click **Next** on the **Welcome** window.
5. On the **Install Prerequisites** window, the appropriate IBM GSKit or IBM Java check box will be selected if required. Click **Next**.
6. On the **Software License Agreement** window, read the agreement and click **Accept**.

7. On the **Select Features** window, select **Tivoli Enterprise Monitoring Server** and click **Next**. Only those components installed on the system are listed.
8. On the **Agent Deployment** window, click **Next**.

   The agent deployment refers to the Internet Service Monitoring application which is assumed to be installed on a separate system.
9. On the **Start Copying Files** window, review the settings and if correct, click **Next**. The files are then copied. A **Setup Status** window informs you about its progress.
10. On the **Setup Type** window, make sure that all setup types are selected and click **Next**.
11. The **Tivoli Enterprise Monitoring Server Configuration** window displays default values. The **TEMS Type**, **TEMS Name** and **Protocol 1** are automatically detected. If required, you can specify a second protocol to be used as a backup. Four types of protocols are available: IP.UDP (uses unsecured UDP communications), IP.PIPE (uses unsecured TCP communications), IP.SPIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
12. Click **OK**.
13. If you have a hub Tivoli Enterprise Monitoring Server, the **Hub TEMS Configuration** window is displayed. The hub settings are automatically detected and depend on the protocol selected in the previous step. The detected settings are:
    - For IP.UDP, the host name or IP address and the Port number (or Port Pools) of the hub server.
    - For IP.PIPE settings, the host name or IP address and the Port number of the hub server (the default port number is 1918).
    - For IP.SPIPE settings, the host name or IP address and the Port number of the hub server (the default port number is 3660).
    - For SNA settings, the SNA network identifier for your location, the LU name for the monitoring server (this LU name corresponds to the Local LU Alias in your SNA Communications software), the name of the LU6.2 LOGMODE (default is CANCTDCS) and the transaction program name.
14. If you have a remote Tivoli Enterprise Monitoring Server, the **Remote TEMS Configuration** window is displayed. Type the name of the remote TEMS in the **Hostname or IP Address** field.
15. In either window, click **OK**. The **Add application support to the TEMS** window is displayed.
16. Select **On this computer** as the location to which the support file should be added and click **OK**. The **Select application support to add to the TEMS** window is displayed.
17. Select **Internet Service Monitoring plugin** and click **OK**.
18. Review the installation summary on the **Application support addition complete** window and click **Next**.
19. Read the README file and click **Finish**.

Results

Installation of Tivoli Enterprise Monitoring Server support is complete.

Installing Tivoli Enterprise Portal Server support on Windows systems

Install Tivoli Enterprise Portal Server (TEPS) support to the system on which Tivoli Enterprise Portal Server is installed.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

To install Tivoli Enterprise Portal Server support on Windows:

1. Log on as a user with administrative privileges.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage/.
3. Launch the installation wizard by double-clicking the `setup.exe` file. The InstallShield Wizard starts.
4. Click **Next** on the **Welcome** window.
5. On the **Install Prerequisites** window, the appropriate IBM GSKit or IBM Java check box will be selected if required. Click **Next**.
6. On the **Software License Agreement** window, read the agreement and click **Accept**.
7. On the **Select Features** window, select **Tivoli Enterprise Portal Server** and click **Next**. Only those components installed on the computer are listed.
8. On the **Agent Deployment** window, click **Next**.

   The agent deployment refers to the Internet Service Monitoring application which is assumed to be installed on a separate system.
9. On the **Start Copying Files** window, review the settings and if correct, click **Next**. The files are then copied. A **Setup Status** window informs you about its progress.
10. On the **Setup Type** window, make sure that all setup types are selected and click **Next**.
11. On the **TEPS Hostname** window, enter the name of the Tivoli Enterprise Portal Server and click **Next**. The installation continues and a **Setup Status** window informs you of its progress.
12. If required, reconfigure the Tivoli Enterprise Monitoring Server connection information on the **Configuration Defaults for Connecting to a TEMS** window and click **OK**.

    Note: The **Configuration Defaults for Connecting to a TEMS** and the subsequent Configuration Defaults for Connecting to a TEMS summary window open only if the Tivoli Enterprise Monitoring Server is installed on the same system. See Chapter 5, "Configuring Internet Service Monitoring," on page 83 for details of the fields on these windows.
13. Read the README file and click **Finish**.

Results

Installation of Tivoli Enterprise Portal Server support is complete.

Installing Tivoli Enterprise Portal support on Windows systems

Install Tivoli Enterprise Portal (TEP) support to the system on which Tivoli Enterprise Portal is installed.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

To install Tivoli Enterprise Portal support on Windows:

1. Log on as a user with administrative privileges.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage/.
3. Launch the installation wizard by double-clicking the `setup.exe` file. The InstallShield Wizard starts.
4. Click **Next** on the **Welcome** window.
5. On the **Install Prerequisites** window, the appropriate IBM GSKit or IBM Java check box will be selected if required. Click **Next**.
6. On the **Software License Agreement** window, read the agreement and click **Accept**.
7. On the **Select Features** window, select **Tivoli Enterprise Portal Desktop Client** and click **Next**. Only those components installed on the current system are listed.
8. On the **Agent Deployment** window, click **Next**.

   The agent deployment refers to the Internet Service Monitoring application which is assumed to be installed on a separate system.
9. On the **Start Copying Files** window, review the settings and if correct, click **Next**. The files are then copied. A **Setup Status** window informs you about its progress.
10. On the **Setup Type** window, make sure that all setup types are selected and click **Next**.
11. On the **TEPS Hostname** window, enter the name of the Tivoli Enterprise Portal Server and click **Next**. The installation continues and a **Setup Status** window informs you of the progress.

    Note: The Tivoli Enterprise Portal Server Hostname is automatically detected if the Tivoli Enterprise Portal Server is installed on the same system.
12. If required, reconfigure the Tivoli Enterprise Monitoring Server connection information on the **Configuration Defaults for Connecting to a TEMS** window and click **OK**.

Note: The **Configuration Defaults for Connecting to a TEMS** and the subsequent summary window are only opened if the Tivoli Enterprise Monitoring Server is installed on the same system. See Chapter 5, "Configuring Internet Service Monitoring," on page 83 for details of the fields on these windows.

13. Read the README file and click **Finish**.

Results

Installation of Tivoli Enterprise Portal support is complete. If all other support files are installed, and the Internet service monitoring agent and Tivoli Enterprise Portal Server are reconfigured if required, you are ready to start Internet Service Monitoring.

Silent installation on Windows systems

Silent installation enables you to define the options for installing Internet Service Monitoring in an installation response file, then run the installation process from the command line without interactive input. This method is useful for performing repeated installations.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

About this task

A sample installation response file, `silent.txt`, is available with the product installer. The file contains comprehensive instructions on how to modify and use it.

Procedure

To run a silent installation:

1. Open the installation response file in a text editor.
2. Uncomment and modify the installation options as required, and then save the file.
3. From a command prompt, change directory to the location of the installer, `setup.exe`, and execute the command:

   ```
   start /wait setup /z"/sfresponse-file" /s /f2"log-file"
   ```

   where *response-file* is the absolute path of the installation response file, and *log-file* is the absolute path to a log file where the installer writes the results of the installation process.

Installing on Linux or UNIX systems

You can install Internet Service Monitoring to a system that also has other IBM Tivoli Monitoring components installed, or you can install Internet Service Monitoring to a separate system.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

To install the Internet Service Monitoring TEMA on a Linux or UNIX system separate from IBM Tivoli Monitoring:

1. Log in as the same user used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the command `./install.sh` and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default, `/opt/IBM/ITM`, or type the full path to a different directory.
6. If the installation directory does not already exist, you are asked if you want to create it. Type 1 to create this directory and press **Enter**.
7. If any existing IBM Tivoli Monitoring components are currently running on the computer, the installer stops them during the installation process, then restarts them when the installation is complete. To confirm this action, type 1 when prompted and press **Enter**. If you choose not to stop the components, the installation process aborts.
8. Type 1 when prompted to `Install products to the local host` and press **Enter**.
9. The software license agreement is displayed. Type 1 to accept the agreement and press **Enter**.
10. If IBM GSKit is not already installed on the computer, you are prompted to provide an encryption key.

    Use the same key across the enterprise. Either type the key or accept the default and press **Enter**.
11. A list is displayed of available operating systems. Type the number for the operating system that you are installing on. The default value is your current operating system. Press **Enter**.
12. Type 1 to confirm the operating system and press **Enter**. A numbered list of products available for installation is displayed.
13. Type the number corresponding to Internet Service Monitoring and press **Enter**.
14. Type 1 to confirm your selection.
15. At the prompt `Do you want to install additional products or product support packages`, type 2 and press **Enter**.

Results

Installation of Internet Service Monitoring is complete. You can now install the support files, then configure the product components. See "Configuring the Internet service monitoring agent on Linux or UNIX systems" on page 84 for more information.

Installing Tivoli Enterprise Monitoring Server support on Linux or UNIX systems

Install Tivoli Enterprise Monitoring Server (TEMS) support to the system on which Tivoli Enterprise Monitoring Server is installed.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

To install Tivoli Enterprise Monitoring Server support on Linux or UNIX:

1. Log in as the same user as that used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the command `./install.sh` and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default, `/opt/IBM/ITM`, or type the full path to a different directory.
6. If the installation directory does not already exist, you are asked if you want to create it. Type `1` to create this directory and press **Enter**.
7. If any existing IBM Tivoli Monitoring components are currently running on the computer, the installer stops them during the installation process, then restarts them when the installation is complete. To confirm this action, type `1` when prompted and press **Enter**. If you choose not to stop the components, the installation process aborts.
8. Type `1` when prompted to `Install products to the local host` and press **Enter**.
9. The software license agreement is displayed. Type `1` to accept the agreement and press **Enter**. A list is then displayed of available operating systems and component support categories.
10. Type the number for Tivoli Enterprise Monitoring Server Support and press **Enter**.
11. Type `1` to confirm your selection and press **Enter**. A list is displayed of products for which support files are to be added.
12. Type the number for Internet Service Monitoring and press **Enter**.
13. Type `1` to confirm your selection and press **Enter**.
14. At the prompt `Do you want to install additional products or product support packages`, type `2` and press **Enter**.

Results

Installation of Tivoli Enterprise Monitoring Server support is complete. You can now add Tivoli Enterprise Monitoring Server application support. See "Adding Tivoli Enterprise Monitoring Server application support on Linux or UNIX systems" for information about how to do this.

Note: If Tivoli Enterprise Portal and Tivoli Enterprise Portal Server are installed on the same system, you can type `1` at the prompt `Do you want to install additional products or product support packages` and press **Enter**. Then follow the prompts to install the Tivoli Enterprise Portal and Tivoli Enterprise Portal Server support files. In a distributed IBM Tivoli Monitoring environment, install the support files separately for each component.

Adding Tivoli Enterprise Monitoring Server application support on Linux or UNIX systems

After you have installed Tivoli Enterprise Monitoring Server support to Linux or UNIX systems, you must add application support.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

About this task

You can add application support from the command line or from the **Manage Tivoli Enterprise Monitoring Services** window.

To add application from the command line, run the command: `./itmcmd support -t` *`temsname`* `is`.

Procedure

To add application support using the **Manage Tivoli Enterprise Monitoring Services** window:

1. Right click Tivoli Enterprise Monitoring Server in the **Manage Tivoli Enterprise Monitoring Services** window.
2. Select **Install Product Support** > **Advanced** and then select **Internet Service Monitoring**.

Installing Tivoli Enterprise Portal Server support on Linux or UNIX systems

Install Tivoli Enterprise Portal Server (TEPS) support to the system on which Tivoli Enterprise Portal Server is installed.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

To install Tivoli Enterprise Portal Server support on Linux or UNIX systems:

1. Login as the same user used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the command `./install.sh` and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default, `/opt/IBM/ITM`, or type the full path to a different directory.
6. If the installation directory does not already exist, you are asked if you want to create it. Type 1 to create this directory and press **Enter**.
7. If any existing IBM Tivoli Monitoring components are currently running on the computer, the installer stops them during the installation process, then restarts them when the installation is complete. To confirm this action, type 1 when prompted and press **Enter**. If you choose not to stop the components, the installation process aborts.
8. Type 1 when prompted to `Install products to the local host` and press **Enter**.
9. The software license agreement is displayed. Type 1 to accept the agreement and press **Enter**. A list is then displayed of available operating systems and component support categories.
10. Type the number for Tivoli Enterprise Portal Server support and press **Enter**.
11. Type 1 to confirm your selection and press **Enter**. A list is displayed of products for which support files are to be added.
12. Type the number for Internet Service Monitoring and press **Enter**.
13. Type 1 to confirm your selection and press **Enter**.
14. At the prompt `Do you want to install additional products or product support packages`, type 1 and press **Enter**.
15. If the system is running Tivoli Enterprise Portal Server, you can elect to install Tivoli Enterprise Portal Browser Client support. From the list of component support categories, type the number for Tivoli Enterprise Portal Browser Client support and press **Enter**.
16. Repeat from step 10 through step 14, selecting Tivoli Enterprise Portal Browser Client Support.
17. At the prompt `Do you want to install additional products or product support packages`, type 2 and press **Enter**.

Results

Installation of Tivoli Enterprise Portal Server support is complete. You can now configure the Tivoli Enterprise Portal Server. See “Configuring Tivoli Enterprise Portal Server on Linux or UNIX systems” on page 85 for further information.

Installing Tivoli Enterprise Portal Desktop support on Linux systems

Install Tivoli Enterprise Portal (TEP) Desktop support to the system on which Tivoli Enterprise Portal is installed.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

Procedure

To install Tivoli Enterprise Portal Desktop support on Linux:

1. Log in as the same user used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the command `./install.sh` and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default, `/opt/IBM/ITM`, or type the full path to a different directory.
6. If the installation directory does not already exist, you are asked if you want to create it. Type 1 to create this directory and press **Enter**.
7. If any existing IBM Tivoli Monitoring components are currently running on the computer, the installer stops them during the installation process, then restarts them when the installation is complete. To confirm this action, type 1 when prompted and press **Enter**. If you choose not to stop the components, the installation process aborts.
8. Type 1 when prompted to `Install products to the local host` and press **Enter**.
9. The software license agreement is displayed. Type 1 to accept the agreement and press **Enter**. A list is then displayed of available operating systems and component support categories.
10. Type the number for Tivoli Enterprise Portal Desktop Client support and press **Enter**.
11. Type 1 to confirm your selection and press **Enter**. A list is displayed of products for which support files are to be added.
12. Type the number for Internet Service Monitoring and press **Enter**.
13. Type 1 to confirm your selection and press **Enter**.
14. At the prompt `Do you want to install additional products or product support packages`, type `n` and press **Enter**.

Results

Installation of Tivoli Enterprise Portal Desktop support is complete. If all other support files are installed, and the Internet service monitoring agent and Tivoli Enterprise Portal Server are configured, you are ready to start Internet Service Monitoring.

Silent installation on UNIX systems

Silent installation enables you to define the options for installing Internet Service Monitoring in an installation response file, then run the installation process from the command line without interactive input. This method is useful for performing repeated installations.

Before you begin

Before beginning the installation, ensure that you have read "Planning to install Internet Service Monitoring" on page 33 for information about hardware and software prerequisites, planning and deployment considerations, and any special limitations.

About this task

A sample installation response file, `silent_install.txt`, is available with the product installer. The file contains comprehensive instructions on how to modify and use it.

Procedure

To run a silent installation:

1. Open the installation response file in a text editor.
2. Uncomment and modify the installation options as required, then save the file.
3. Execute the command:

   ```
   ./install.sh -q -h home -p response-file
   ```

   where *home* is the IBM Tivoli Monitoring home directory, typically `/opt/IBM/ITM`, and *response-file* is the absolute pathname of the installation response file.

Results

What to do next

After installation is complete, you can configure the product using the silent configuration process.

Silent configuration

After installing Internet Service Monitoring, you can configure it silently.

About this task

To perform silent configuration, you must create a configuration response file. Configuration response files are text files containing parameter-value pairs that specify the desired configuration settings.

The following syntax rules apply to configuration response files:

- Comment lines begin with a pound (#) character.
- Blank lines are ignored.
- Parameter-value pairs have the format: *PARAMETER=value*

  Do not use a space before the parameter name; you can use a space before or after an equal (=) character.

Do not use dollar ($), equal (=), or pipe (|) characters in parameter values.

Procedure

To perform silent configuration:

1. Create a configuration response file containing the desired configuration settings.

   See "Sample silent configuration file" for an example.
2. Execute the command:

   ```
   ./itmcdm config -A -p response-file is
   ```

   where *response-file* is the absolute path to the configuration response file.

Sample silent configuration file

```
################## PRIMARY TEMS CONFIGURATION ##################

# Will this agent connect to a Tivoli Enterprise Monitoring Server (TEMS)?
# This parameter is required.
# Valid values are: YES and NO
##### TEP should not connect to TEMS, ignore all TEMS connection variables. #####
#CMSCONNECT=YES

# What is the hostname of the TEMS to connect to?
# This parameter is NOT required.  (default is the local system hostname)
#HOSTNAME=somehost.somewhere.com

# Will this agent connect to the TEMS through a firewall?
# This parameter is NOT required.  (default is NO)
# Valid values are: YES and NO
#   - If set to YES the NETWORKPROTOCOL must be ip.pipe
#FIREWALL=NO

# What network protocol is used when connecting to the TEMS?
# This parameter is required.
# Valid values are: ip, sna, ip.pipe, or ip.spipe
#NETWORKPROTOCOL=ip.pipe

# What is the first backup network protocol used for connecting to the TEMS?
# This parameter is NOT required. (default is none)
# Valid values are: ip, sna, ip.pipe, ip.spipe, or none
#BK1NETWORKPROTOCOL=none

# What is the second backup network protocol used for connecting to the TEMS?
# This parameter is NOT required. (default is none)
# Valid values are: ip, sna, ip.pipe, ip.spipe or none
#BK2NETWORKPROTOCOL=none

# If ip.pipe is one of the three protocols what is the IP pipe port number?
# This parameter is NOT required. (default is 1918)
#IPPIPEPORTNUMBER=1918

# If ip.pipe is one of the three protocol what is the IP pipe partition name?
# This parameter is NOT required. (default is null)
#KDC_PARTITIONNAME=null

# If ip.pipe is one of the three protocols what is the KDC partition file?
# This parameter is NOT required. (default is null)
#KDC_PARTITIONFILE=null

# If ip.spipe is one of the three protocols what is the IP pipe port number?
# This parameter is NOT required. (default is 3660)
#IPSPIPEPORTNUMBER=3660
```

```
# If ip is one of the three protocols what is the IP port number?
# This parameter is NOT required. (default is 1918)
# A port number and or one or more pools of port numbers can be given.
# The format for a pool is #-# with no embedded blanks.
#PORTNUMBER=1918

# If sna is one of the three protocols what is the SNA net name?
# This parameter is NOT required. (default is CANDLE)
#NETNAME=CANDLE

# If sna is one of the three protocols what is the SNA LU name?
# This parameter is NOT required. (default is LUNAME)
#LUNAME=LUNAME

# If sna is one of the three protocols what is the SNA log mode?
# This parameter is NOT required. (default is LOGMODE)
#LOGMODE=LOGMODE

################## SECONDARY TEMS CONFIGURATION #################

# Would you like to configure a connection for a secondary TEMS?
# This parameter is NOT required. (default is NO)
# Valid values are: YES and NO
#FTO=NO

# If configuring a connection for a secondary TEMS, what is the hostname
# of the secondary TEMS?
# This parameter is required if FTO=YES
#MIRROR=somehost.somewhere.com

# Will the TEP connect to the secondary TEMS through a firewall?
# This parameter is NOT required. (default is NO)
# Valid values are: YES and NO
#FIREWALL2=NO

# What network protocol is used when connecting to the secondary TEMS?
# This parameter is required when FTO=YES and FIREWALL2 is NO
# Valid values are: ip, sna, or ip.pipe
#HSNETWORKPROTOCOL=ip.pipe

# What is the first backup network protocol used for connecting to the
# secondary TEMS?
# This parameter is NOT required. (default is none)
# Valid values are: ip, sna, ip.pipe, or none
#BK1HSNETWORKPROTOCOL=none

# What is the second backup network protocol used for connecting to the
# secondary TEMS?
# This parameter is NOT required. (default is none)
# Valid values are: ip, sna, ip.pipe, or none
#BK2HSNETWORKPROTOCOL=none

# If ip.pipe is one of the three secondary TEMS protocols what is the IP pipe
# port number?
# This parameter is NOT required. (default is 1918)
#HSIPPIPEPORTNUMBER=1918

# If ip is one of the three secondary TEMS protocols what is the IP port number?
# This parameter is NOT required. (default is 1918)
# A port number and or one or more pools of port numbers can be given.
# The format for a pool is #-# with no embedded blanks.
#HSPORTNUMBER=1918

# If sna is one of the three secondary TEMS protocols what is the SNA net name?
# This parameter is NOT required. (default is CANDLE)
#HSNETNAME=CANDLE
```

```
# If sna is one of the three secondary TEMS protocols what is the SNA LU name?
# This parameter is NOT required. (default is LUNAME)
#HSLUNAME=LUNAME

# If sna is one of the three secondary TEMS protocols what is the SNA log mode?
# This parameter is NOT required. (default is LOGMODE)
#HSLOGMODE=LOGMODE

################## OPTIONAL PRIMARY NETWORK NAME CONFIGURATION #################

# If the system is equipped with dual network host adapter cards you can designate
# another network name.  What is the network name?
# This parameter is NOT required. (default is none)
#PRIMARYIP=none

################## ISM CONFIGURATION #################

# Connect ISMs to an Object Server
# This parameter is required.(default is no)
#ISM_OMNIBUS_CONNECTION=no

# What is the Object server hostname?
# (default is localhost)
#ISM_OMNIBUS_HOSTNAME=localhost

# What is the Object Server port?
# (default is 4100)
#ISM_OMNIBUS_PORT=4100

# What is the name of the Object Server?
# (default is "NCOMS")
#ISM_OMNIBUS_NAME="NCOMS"
```

Upgrading Internet Service Monitoring

Upgrade Internet Service Monitoring components to the latest version using an installer. The installer updates Internet Service Monitoring and its support files. Run the installer on each computer that has Internet Service Monitoring installed.

Before you begin

For information about obtaining the latest version, see the *Download information* on the ITCAM for Transactions Information Center.

Note: If you are upgrading Internet Service Monitoring V7.1 to a higher fix pack, first ensure that you save any customized rules or properties files to a temporary directory. By default,ISMHOME directory is:

- For Windows systems, `C:\IBM\ITM\TMAITM6\ism\`
- For UNIX systems, `/opt/IBM/ITM/`*`itm_arch`*`/is/`

After you have saved your custom files, upgrade Internet Service Monitoring V7.1. When the upgrade is complete, restore your custom files from your backup. This is not necessary when upgrading to V7.2 or later because custom files are preserved automatically.

Note: Ensure that you upgrade all of the following components in IBM Tivoli Composite Application Manager for Transactions to the latest release on all relevant computers:

- Internet Service Monitoring
- Tivoli Enterprise Monitoring Server support

- Tivoli Enterprise Portal Server support
- Tivoli Enterprise Portal support

About this task

This procedure describes how to upgrade Internet Service Monitoring on Windows. The procedure is similar for the other operating systems and similar to the full installation.

Procedure

To upgrade Internet Service Monitoring for all components installed on a particular computer:

1. Log on as a user with administrative privileges.
2. Stop the Databridge and monitor services. (On UNIX systems, also stop the processes.)
3. From the DVD or downloaded and extracted package, double-click `setup.exe` to start the installation wizard.
4. On the **Welcome** window, click **Next**.
5. On the **Install Prerequisites** window, options to ensure that you have the correct version of IBM GSKit or IBM Java are selected. Click **Next**. The required software is installed automatically.
6. On the **Software License Agreement** window, read the agreement and click **Accept**.
7. On the **Select Features** window, the components installed on the computer are selected. Expand **Tivoli Enterprise Monitoring Agents**, ensure all appropriate options are selected, and click **Next**.
8. On the **Agent Deployment** window, select **Internet Service Monitoring** if you want to deploy to a remote location by using the Tivoli Enterprise Monitoring Server and click **Next**. If you are installing locally, do not select any agents.

   See Chapter 11, "Working remotely," on page 309 for further information.
9. On the **Start Copying Files** window, review the settings and if correct, click **Next**. Click **Yes** in the confirmation dialog box to start copying files. A **Setup Status** window informs you about the progress of the installation.
10. On the **Setup Type** window, select all configuration options and click **Next**.

    You can delay some configuration until after installation if required. The following steps assume that all configuration options are selected.
11. On the **Configuration Defaults for Connecting to a TEMS** window and subsequent summary window, check the settings and click **OK**.
12. Internet Service Monitoring is upgraded, configured, and started. The **InstallShield Wizard Complete** window opens when installation is complete. Click **Finish** and read the `readme` file.

What to do next

After upgrading Internet Service Monitoring on UNIX systems, re-configure the agent to reestablish connection to Tivoli Enterprise Monitoring Server.

Use the Manage Tivoli Enterprise Monitoring Services console or the command `itmcmd config -A is`. See Chapter 5, "Configuring Internet Service Monitoring," on page 83 for further information.

Update Internet Service Monitoring components installed on other computers to the latest Internet Service Monitoring version.

Ensure that you also upgrade Tivoli Enterprise Monitoring Server support, Tivoli Enterprise Portal Server support, and Tivoli Enterprise Portal support.

If your existing profiles were created using **ismbatch** and you now want to view the monitor results and edit the profiles in the Tivoli Enterprise Portal, import the profiles using **ismconfig**. To import the profiles:

- For Windows systems, run `ismconfig.cmd -file filename`
- For UNIX systems, run `ismconfig.sh -file filename`

where *filename* indicates a text file containing ismbatch commands. For example:

```
-add monitor=http profile=testprofile server=testserver1
-add monitor=http profile=testprofile server=testserver2
-add monitor=icmp profile=testprofile server=testserver3
```

After the profiles are imported they can be managed in the Tivoli Enterprise Portal.

Important: After upgrading, you may need to stop and restart all situations on the Tivoli Enterprise Portal to ensure that situation action commands are sent correctly and the bridge and monitors are started.

Upgrading Internet Service MonitoringV2.4R2 and V6.0.0

Upgrade Internet Service Monitoring versions 2.4R2, and 6.0.0 by installing Internet Service Monitoring V7.3 and migrating the required information.

Before you begin

To upgrade an earlier installation of Internet Service Monitoring, install Internet Service Monitoring V7.3. The necessary updates occur automatically.

If required, after installing Internet Service Monitoring V7.3 you can migrate profiles, properties files, and rules files from an existing Netcool/ISM version 2.4R2, or 6.0.0 installation. You can then start monitoring with Internet Service Monitoring V7.3. You do not have to manually migrate profiles, properties files, or rules files when upgrading Internet Service Monitoring V6.0.1 or later.

Procedure

To migrate data from an existing installation to Internet Service Monitoring V7.3:

1. Convert existing profiles.

   Internet Service Monitoring V7.3 provides tools for converting existing profiles.

   Note: In these guidelines, `ISMHOME` refers to the V7.3 location.

   To convert profiles:

   a. Create the directory `$ISMHOME/ismbackup/profiles` and copy any profiles from the existing installation that you want to convert to this directory.
   b. Create the directory `$ISMHOME/ismbackup/profiles/default` and copy the files `oidgroups.xml` and `profile.xsd` from the existing installation to this directory.
   c. Convert the profiles to V7.3 format.

      On Linux and UNIX systems, execute the command:

```
$ISMHOME/bin/profileUpgrade
```

On Windows, execute the command:

```
%ISMHOME%\platform\win32\bin\profileUpgrade.exe
```

The converted profiles are placed in the directory `$ISMHOME/profiles/active`. If required, you may specify different locations for the profile inputs using the following switches:

- `-profile` sets the input directory
- `-profile24defaults` sets the path to the `oidgroups.xml` and `profile.xsd` files.

The `-messagelevel` and `-messagelog` switches control the debug information generated by the profile upgrade tool.

2. Transfer custom properties.

   V7.3 properties files contain new properties, and modify some properties that existed in previous product releases. If you want to retain specific property settings from an existing installation, add them to the V7.3 files manually. Overwriting V7.3 properties files with those from the existing installation is not recommended.

3. Copy custom rules files.

   If you have modified any rules files and want to retain those changes, copy the rules files from the existing installation to the V7.3 installation.

   Note: V7.3 includes two monitors not available in versions earlier than V7.1, SIP and SOAP. If you copy an existing `objectserver.rules` file to the installation, ensure that you add `include` statements for the SIP and SOAP monitor rules files.

What to do next

If you have installed Internet Service Monitoring onto an IBM Tivoli Monitoring environment, import the profiles for use in Tivoli Enterprise Portal:

1. Convert the profiles to ismconfig operations by using the xml2cli application. For example:

   ```
   xml2cli > ismprofiles.txt
   ```

   See Converting profiles created with ismbatch to ismconfig operations in the Administrator's Guide for further information.

2. Import information into the Tivoli Enterprise Portal Server database by using the `ismconfig -config -file` command. For example,

   ```
   ismconfig -config -file ismprofiles.txt
   ```

   See the Internet Service Monitoring Configuration command-line interface in the Administrator's Guide for further information.

Starting Internet Service Monitoring

Starting Internet Service Monitoring requires you to start each of the product components individually.

Before you begin

These guidelines assume that you have already started the Tivoli Enterprise Monitoring Server (TEMS) and the Tivoli Enterprise Portal Server (TEPS).

Procedure

To start Internet Service Monitoring:

1. *(Optional)* Start the Databridge and the Internet service monitors.
2. Start the Internet service monitoring agent.

   Note: As of ITCAM for Transactions V7.2, starting and stopping the Internet Service Monitoring agent also starts and stops all Internet service monitors and the Databridge automatically.
3. To create Internet service tests and view their results, start the Tivoli Enterprise Portal client.

Starting Internet Service Monitoring on Windows systems

To start Internet Service Monitoring on Windows, follow these steps.

About this task

To start the Databridge and the Internet service monitors:

1. Select **Start** > **Control Panel**, open the **Administrative Tools** group, then launch the **Services** console.
2. From the list of services, select the service named **NCO BRIDGE Internet Service Monitor**, and then select **Start** from the context menu.

   Tip: The installer configures the Databridge service to start automatically, so it is normally not necessary to start it manually.
3. From the list of services, select the service named **NCO** ***service-name*** **Internet Service Monitor** for each Internet service monitor that you want to run, then select **Start** from the context menu.

   Note: If you have deployed Internet service monitors on more than one host, you must start the monitor services on each host.

To start the Internet service monitoring agent:

1. Select **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
2. From the list of components, select **Internet Service Monitoring**, and then select **Start** from the context menu.

To start the Tivoli Enterprise Portal desktop client:

1. Select **Start** > **Programs** > **IBM Tivoli Monitoring** > **Tivoli Enterprise Portal**. Alternatively, if you created a desktop icon, click the icon. The **Logon** window is displayed.
2. Enter your logon ID and password and click **OK**.

Alternatively, to start the Tivoli Enterprise Portal browser client:

1. In the **Address** field in Internet Explorer, enter `http://`*`ip of Tivoli Enterprise Portal Server`*`:1920`. For example, http://192.168.60.22:1920.
2. In the **IBM Tivoli Monitoring Service Index**, select **IBM Tivoli Enterprise Portal Web Client**.
3. In the **Warning - Security** dialog box, click **Yes** to indicate that you trust the applet distributed by IBM. If you do not trust the applet you will not be able to use the browser client.
4. Enter your user name and password and click **OK**.

The Internet Service Monitoring user configuration application is indicated in the toolbar of the Tivoli Enterprise Portal by its  icon.

Note: When you start the user interface configuration, the **Internet Service Monitoring Configuration Loading** window is displayed. Closing this window while the interface configuration is still being loaded, does not stop the interface configuration from being launched. If you do want to stop using the configuration, wait until it is loaded.

Starting Internet Service Monitoring on Linux or UNIX systems

To start Internet Service Monitoring on Linux or UNIX systems, follow these steps.

About this task

To start the Databridge and the Internet service monitors, run the command:

```
$ISMHOME/bin/ism_startup.sh start
```

To start the Internet service monitoring agent, run the command:

```
/opt/IBM/ITM/bin/itmcmd agent start is
```

To start the Tivoli Enterprise Portal desktop client:

1. Run the command:

   ```
   /opt/IBM/ITM/bin/itmcmd agent start cj
   ```

2. Enter your login ID and password, and then click **OK**.

Alternatively, to start the Tivoli Enterprise Portal browser client:

1. In the **Address** field in Internet Explorer, enter `http://`*`ip of Tivoli Enterprise Portal Server`*`:1920`. For example, http://192.168.60.22:1920.
2. In the **IBM Tivoli Monitoring Service Index**, select **IBM Tivoli Enterprise Portal Web Client**.
3. In the **Warning - Security** dialog box, click **Yes** to indicate that you trust the applet distributed by IBM. If you do not trust the applet you will not be able to use the browser client.
4. Enter your user name and password and click **OK**.

The Internet Service Monitoring application is indicated in the toolbar of the Tivoli Enterprise Portal by the Internet Service Monitoring Configuration icon  .

Note: When you start the user interface configuration, the **Internet Service Monitoring Configuration Loading** window is displayed. Closing this window while the interface configuration is still being loaded, does not stop the interface configuration from being launched. If you do want to stop using the configuration,

wait until it is loaded.

Starting and stopping Internet Service Monitoring monitors using Tivoli Enterprise Portal

You can use Tivoli Enterprise Portal to start and stop monitors and the Databridge, including those deployed on remote computers.

Before you begin

To control monitors and the Databridge from the Tivoli Enterprise Portal, a Universal Agent must already be installed on the computer hosting those components.

Procedure

To start or stop the Internet Service Monitoring monitors using Tivoli Enterprise Portal:

1. Log into Tivoli Enterprise Portal.
2. In the Navigator pane, locate the computer hosting the monitors and expand its node.
3. Select **Internet Service Monitors**, and then right-click and select **Take Action** > **Select**.
4. In the **Take Action** dialog box, select the required action from the **Action** list. For example, to start the DHCP monitor, select **Start DHCP**.
   - You can enter specific combinations of monitors in the **Command** field, for example `start DHCP FTP`.
   - To start or stop all components, select the **Start all** or **Stop all** actions.
5. In the **Destinations** group, select all the host computers on which to perform the action.
6. Click **OK**.

What to do next

The **Action Status** dialog box displays the results of the selected actions. A `Return Code` of zero indicates that the actions were successful. The `Message` field provides additional information, where applicable.

Stopping Internet Service Monitoring

In ITCAM for Transactions V7.2 and later, when you stop the Internet Service Monitoring agent, you stop all monitors and the Databridge by default.

Before you begin

You can stop the Internet Service Monitoring agent by using the Manage Tivoli Enterprise Monitoring Services, or from the command line with `tacmd` on Windows systems or `itmcmd` on UNIX systems.

About this task

If you do not want the Internet Service Monitoring agent to automatically start and stop all monitors and the Databridge, you can disable the **`ManageServices`** property in the Internet Service Monitoring properties file (`kisagent.props`) or by using the command line.

Note: Disabling the **ManageServices** property also disables Take Actions commands and situations from starting the monitors and Databridge.

Procedure

To disable the **ManageServices** property:

1. In `kisagents.props`, set the **ManageServices** property to `0`.
2. Or from the command line, run `tacmd -manageservices 0` using `tacmd` on Windows systems or `itmcmd` on UNIX systems.
3. To enable the property again, set the value back to `1`.

Troubleshooting the Internet Service Monitoring installation

If you are experiencing difficulties after installing Internet Service Monitoring check here first to help you resolve any problems.

- **The Internet Service Monitoring Configuration icon does not appear in the Tivoli Enterprise Portal.**

  If the Internet Service Monitoring Configuration icon  does not appear in the Tivoli Enterprise Portal toolbar when Internet Service Monitoring installation is complete, do the following:

  – Check that Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal desktop or Tivoli Enterprise Portal browser support are installed for Internet Service Monitoring on the appropriate machines

  – Reconfigure the Tivoli Enterprise Portal Server using the command line or Manage Tivoli Enterprise Monitoring Services

  – Clear the Java cache using the Windows Control Panel

  – If using the Tivoli Enterprise Portal browser, clear the web browser cache

- **A node running the Internet Service Monitors is not displayed in the Navigator.**

  If the node is not visible at all, the connection between the Internet Service Monitoring Agent and the Tivoli Enterprise Monitoring Server is not configured correctly. Reconfigure the Internet Service Monitors on that node.

- **The Internet Service Monitoring workspaces are not displayed in the Navigator.**

  If the Internet Service Monitoring workspaces are not visible at all, check that you have installed the Tivoli Enterprise Portal Server support files on the computer running Tivoli Enterprise Portal Server.

- **The Internet Service Monitoring workspaces are displayed in the Navigator, but have unusual names.**

  If the workspaces are visible, but have names starting with KIS, check that you have installed the Tivoli Enterprise Portal Desktop Client support files on the computer running the client.

- **The Internet Service Monitoring workspaces are not available.**

  If the workspaces are visible but not available, the Internet service monitoring agent has run in the past but conditions have changed: either the agent is not running now or the connection information to the Tivoli Enterprise Monitoring Server has changed.

- **When does polling start and when should I see data in the workspaces?**

  Polling starts when the Internet service monitoring agent starts and at every poll interval specified by the profile element. If you do not see any data, check the poll interval and check that the Databridge and monitors are running.

If you installed on a distributed system, check that you installed the correct support files on each computer.

- **There is no data in the history workspaces**

  Check that you have configured the historical data collection for the data source. In addition, if using the Tivoli Data Warehouse for long-term reporting, check that you have configured the pruning and summarization of the data.

  If the Internet service monitoring agent is active and historical data is configured, but there is still no data in the history workspaces, check that the Internet service monitoring agent is running.

Uninstalling Internet Service Monitoring

Uninstalling Internet Service Monitoring is a two-step process if installed on the same system that is host to Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal. You must first uninstall the product from the system and then remove the product from the Tivoli Enterprise Portal into which it is integrated.

About this task

If Internet Service Monitoring is installed in a distributed IBM Tivoli Monitoring environment, an additional step is required to remove the Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal support files.

Removing Internet Service Monitoring does not affect your IBM Tivoli Monitoring

Uninstalling Internet Service Monitoring on Windows systems

Uninstalling Internet Service Monitoring from a Windows system that is also host to Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server automatically removes the associated support files. If you have a distributed installation, ensure that you uninstall the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server support files separately from the remote systems.

Before you begin

Before uninstalling Internet Service Monitoring on Windows, run the pre-installation script `preuninstall.cmd` to clean up the services that were created during installation. The script is located in `%CANDLE_HOME%\TMAITM6\ism\platform\win32\bin\preuninstall.cmd`.

When you have run the script, you can uninstall Internet Service Monitoring.

Procedure

To uninstall Internet Service Monitoring on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add and Remove Programs**.
2. Navigate to IBM Tivoli Monitoring and click **Remove**.
3. On the **Welcome** window of the IBM Tivoli Monitoring installation wizard, select **Modify** to remove only the selected features, or select **Remove** to remove all features and click **Next**.
4. Click **OK** in the information dialog box.

5. On the **Add or Remove Features** window, deselect those features you want to uninstall and click **Next**.
6. Click **OK** in the confirmation dialog box. Internet Service Monitoring is removed.
7. On the **Product Remove Complete** window, click **Finish**.

Results

Internet Service Monitoring is removed. If you want to clear the corresponding Internet Service Monitoring agents from display in the Tivoli Enterprise Portal, you must do this manually. See "Removing agents from Tivoli Enterprise Portal."

Uninstalling Internet Service Monitoring on UNIX systems

Uninstalling Internet Service Monitoring from a UNIX system that is also host to Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server, automatically removes the associated support files. If you have a distributed installation, ensure that you uninstall the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server support files separately from the remote systems.

Procedure

To uninstall Internet Service Monitoring on UNIX systems:

1. From a command shell, type `cd /opt/IBM/ITM/bin`.
2. Type the command `./uninstall.sh`. The name of the system on which the product is installed, is displayed.
3. Type the number corresponding to Internet Service Monitoring (this is the entry indicated by the product code `is` *`os`*).
4. Enter `1` to confirm your selection and press **Enter**.
5. To exit the uninstaller, enter `99` and press **Enter**.

Results

Internet Service Monitoring is removed. If you want to clear the corresponding Internet Service Monitoring agents from display in the Tivoli Enterprise Portal, you must do this manually. See "Removing agents from Tivoli Enterprise Portal."

Removing agents from Tivoli Enterprise Portal

After you have uninstalled Internet Service Monitoring, you may want to clear unused Internet Service Monitoring agents from display in the Tivoli Enterprise Portal (TEP).

Procedure

To clear an unused Internet Service Monitoring agent from Tivoli Enterprise Portal:

1. In the Tivoli Enterprise Portal, select **Enterprise** in the navigator.
2. Right click on **Enterprise** and select **Workspace** > **Managed System Status**.
3. In the **Managed System Status** view, right click Internet Service Monitoring that has a status of **Offline** and select **Clear offline entry**. The application is removed from the Tivoli Enterprise Portal.
4. Click **Refresh** to update the Tivoli Enterprise Portal display.

Uninstalling support files

In an IBM Tivoli Monitoring environment where both Internet Service Monitoring and the IBM Tivoli Monitoring components are installed on the same system, the support files are automatically removed when uninstalling Internet Service Monitoring. In a distributed environment, you must manually remove the support files from the remote systems.

Uninstalling Tivoli Enterprise Monitoring Server support on Windows systems

Uninstall both Tivoli Enterprise Monitoring Server (TEMS) application support and the support file from Tivoli Enterprise Monitoring Server.

About this task

Remove Tivoli Enterprise Monitoring Server application support first and then remove the Tivoli Enterprise Monitoring Server support component.

To remove Tivoli Enterprise Monitoring Server application support on Windows:

1. Open the **Manage Tivoli Enterprise Monitoring Services** window on the system where Tivoli Enterprise Monitoring Server is installed.
2. Right click **Tivoli Enterprise Monitoring Server**.
3. Select **Advanced** > **Remove TEMS application support**.
4. In the **Remove application support from the TEMS** dialog box, select **On this computer** and click **OK**.
5. In the **Select the application support to remove from the TEMS** dialog box, select **Internet Service Monitoring** and click **OK**.

Procedure

To uninstall Tivoli Enterprise Monitoring Server support on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add and Remove Programs**.
2. Navigate to **IBM** and click **Change/Remove**.
3. On the **Welcome** window of the Internet Service Monitoring installation wizard, select **Modify** and click **Next**.
4. On the **Add or Remove Features** window, expand **Tivoli Enterprise Monitoring Server**, deselect **Internet Service Monitoring** and click **Next**.
5. Follow the prompts to complete the uninstall process and click **Finish**.

Uninstalling Tivoli Enterprise Portal Server support on Windows systems

Uninstall Tivoli Enterprise Portal Server (TEPS) support from Tivoli Enterprise Portal Server.

Procedure

To uninstall Tivoli Enterprise Portal Server support on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add and Remove Programs**.
2. Navigate to IBM Internet Service Monitoring and click **Change/Remove**.
3. On the **Welcome** window of the Internet Service Monitoring installation wizard, select **Modify** and click **Next**.
4. On the **Add or Remove Features** window, expand **Tivoli Enterprise Portal Server**, deselect **Internet Service Monitoring** and click **Next**.

5. Follow the prompts to complete the uninstall process and click **Finish**.

Uninstalling Tivoli Enterprise Portal support on Windows systems

Uninstall Tivoli Enterprise Portal (TEP) support by uninstalling both Tivoli Enterprise Portal Browser Client support and Tivoli Enterprise Portal Desktop Client support from Tivoli Enterprise Portal.

About this task

On Windows, Tivoli Enterprise Portal Browser Client support is bundled with Tivoli Enterprise Portal Server support. The browser client is uninstalled when Tivoli Enterprise Portal Server support is uninstalled.

Procedure

To uninstall Tivoli Enterprise Portal Desktop Client support on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add and Remove Programs**.
2. Navigate to Internet Service Monitoring and click **Change/Remove**.
3. On the **Welcome** window of the Internet Service Monitoring installation wizard, select **Modify** and click **Next**.
4. On the **Add or Remove Features** window, expand **Tivoli Enterprise Portal Desktop Client**, deselect **Internet Service Monitoring** and click **Next**.
5. Follow the prompts to complete the uninstallation process and click **Finish**.

Reinstalling Internet Service Monitoring

Before reinstalling Internet Service Monitoring check here.

To reinstall Internet Service Monitoring follow the procedures described in Chapter 4, “Installing Internet Service Monitoring,” on page 55.

Chapter 5. Configuring Internet Service Monitoring

Before you can use Internet Service Monitoring, it must be configured to work with IBM Tivoli Monitoring or IBM Tivoli Netcool/OMNIbus.

Configuring the Internet service monitoring agent on Windows systems

To configure the Internet service monitoring agent to work with IBM Tivoli Monitoring, you must set connection parameters that enable it to contact the Tivoli Enterprise Monitoring Server (TEMS), or the ObjectServer if using IBM Tivoli Netcool/OMNIbus. On Windows, these parameters are typically configured during installation of the Tivoli Enterprise Monitoring Server support file but you can reconfigure them at any time by starting the configuration procedure manually.

About this task

Each configuration generates a log file that you can use to diagnose problems. The location of the log file is

```
\opt\IBM\ITM\InstallITM\plugin\ExecutionEvents\logs\
install_plugin_comp_is_n
```

where *n* is a number increasing by one for each configuration.

Note: Ensure that Tivoli Enterprise Portal Server is running before you configure Internet Service Monitoring.

Procedure

To manually configure the Internet Service Monitoring agent on Windows systems:

1. Select **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
2. Right-click **Internet Service Monitoring** and select **Reconfigure**. The **Agent Advanced Configuration** window opens. This window is called **Configuration Defaults for Connecting to a TEMS** if accessed during the installation process.
3. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:
   a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.
   b. Select **Protocol 1** and select a protocol from the list. Several types of protocols are available: IP.UDP (uses unsecured UDP communications), IP.PIPE (uses unsecured TCP communications), IP.SPIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
   c. If additional protocols are required, select **Protocol 2** and select an second protocol from the list.
   d. Do not select **Optional Secondary TEMS Connection**. You can set up the failover support for the component later. See the *IBM Tivoli Monitoring User's Guide* for further information.
   e. Click **OK**.

4. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**.

   These settings define communications between the agent and the Tivoli Enterprise Monitoring Server. The host name or IP address of the local computer are displayed unless the Tivoli Enterprise Monitoring Server has already been specified. Ensure that you enter the host name or IP address of the Tivoli Enterprise Monitoring Server in the **Hostname or IP address** field if it is installed on another computer. The default port number for the previously selected protocol is also displayed (IP.PIPE is 1918, IS.SPIPE is 3660).
5. On the **Internet Service Monitoring Configuration** window, if using IBM Tivoli Netcool/OMNIbus select **YES** and enter the host name or IP address of the IBM Tivoli Netcool/OMNIbus ObjectServer and its port number.

   Tip: If you are using IBM Tivoli Netcool/OMNIbus but want to configure the connection later, select **NO**. Configure the ObjectServer connection later using the Manage Tivoli Enterprise Monitoring Services.
6. Click **OK**. The system continues the agent configuration. Upon completion you are returned to the **Manage Tivoli Monitoring Services** window.

Configuring the Internet service monitoring agent on Linux or UNIX systems

To configure the Internet service monitoring agent to work with IBM Tivoli Monitoring, you must set the connection parameters that enable it to contact the Tivoli Enterprise Monitoring Server (TEMS) or to the ObjectServer if using IBM Tivoli Netcool/OMNIbus. This configuration must be done manually on Linux or UNIX systems after installing Internet Service Monitoring.

About this task

Perform the configuration as the same user that was used when IBM Tivoli Monitoring was installed. You can update your configuration at any time. A log file is generated for each configuration. Use this file to diagnose any problems.

The file is located at:

```
/opt/IBM/ITM/InstallITM/plugin/executionEvents/logs/
install_plugin_comp_is_n
```

where *n* is a number increasing by one for each generation.

Note: Ensure that Tivoli Enterprise Portal Server is running before you configure the Internet service monitoring agent.

To configure the Internet service monitoring agent from the command line, run the following command from the computer where the agent is installed:

```
/opt/IBM/ITM/bin/itmcmd config -A is
```

Procedure

To configure the agent using **Manage Tivoli Enterprise Monitoring Services**:

1. Change directory to `/opt/IBM/ITM/bin`.
2. Run the command: `./itmcmd manage`. The **Manage Tivoli Enterprise Monitoring Services** window is displayed.

3. Select **Internet Service Monitoring**.
4. Right click and select **Configure**. The **Internet Service Monitoring Configuration** window is displayed, and the **Object Server Connection** tab is presented.
5. If using Netcool/OMNIbus, set **Configure Object Server Connection** to **YES** and enter the name of the ObjectServer, the host name or IP address of the Netcool/OMNIbus ObjectServer computer and its port number. Click **OK**.
6. Ensure that the check box for **No TEMS** is cleared and type the **TEMS Hostname**.
7. On the **Protocol 1** tab select the protocol required to communicate with the monitoring server. You can specify two protocols, one as the standard protocol and one to be used as a backup. Four types of protocol are available: IP.TCP, IP.PIPE (uses unsecured TCP communications), IS.PIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
8. Enter the settings or accept the defaults for the selected protocol and click **Save**.

   Note: The default port number for IP.PIPE is 1918, for IP.SPIPE the port number is 3660.

Configuring Tivoli Enterprise Portal Server on Linux or UNIX systems

After you have installed Tivoli Enterprise Portal Server (TEPS) Support on Linux or UNIX systems you must reconfigure the Tivoli Enterprise Portal Server.

Before you begin

You can reconfigure Tivoli Enterprise Portal Server after you have completed the installation of Tivoli Enterprise Portal Server Support using the command line or **Manage Tivoli Enterprise Monitoring Services**.

To reconfigure Tivoli Enterprise Portal Server from the command line:

1. Run the following command on the computer where the Tivoli Enterprise Portal Server is installed:

   ```
   /opt/IBM/ITM/bin/itmcmd config -A cq
   ```

2. (Linux only) Run the following command on the computer where the Tivoli Enterprise Portal Desktop Client is installed:

   ```
   /opt/IBM/ITM/bin/itmcmd config -A cj
   ```

Procedure

To reconfigure the Tivoli Enterprise Portal Server using **Manage Tivoli Enterprise Monitoring Services**:

1. Open the **Manage Tivoli Enterprise Monitoring Services** window on the computer where the Tivoli Enterprise Portal Server is installed.
2. Right click **Tivoli Enterprise Portal Server** and select **Configure**.
3. In the **Configure** dialog box, select **Save** (no changes are required).
4. (Linux only) Open the **Manage Tivoli Enterprise Monitoring Services** window on the computer where the Tivoli Enterprise Portal Desktop Client is installed.
5. Right click **Tivoli Enterprise Portal Desktop Client** and select **Configure**.
6. In the **Configure** dialog box, select **Save** (no changes are required).

Configuring Tivoli Enterprise Portal on Linux systems

After you have installed Tivoli Enterprise Portal Desktop Support on a Linux system you must reconfigure the Tivoli Enterprise Portal. You can reconfigure the Tivoli Enterprise Portal Desktop Client after you have completed the installation of Tivoli Enterprise Portal Desktop Support using the command line or **Manage Tivoli Enterprise Monitoring Services**.

Before you begin

To reconfigure Tivoli Enterprise Portal Desktop Client from the command line, run the following command on the computer where the Tivoli Enterprise Portal Desktop Client is installed:

```
/opt/IBM/ITM/bin/itmcmd config -A cj
```

Procedure

To reconfigure the Tivoli Enterprise Portal Desktop Client Tivoli Enterprise Portal Server using **Manage Tivoli Enterprise Monitoring Services**:

1. Open the **Manage Tivoli Enterprise Monitoring Services** window on the computer where the Tivoli Enterprise Portal Desktop Client is installed.
2. Right-click **Tivoli Enterprise Portal Desktop Client** and select **Configure**.
3. In the **Configure** dialog box, select **Save** (no changes are required).

Databridge configuration

Configuring the Databridge involves setting properties for the Databridge that control its operation such as the connection of the component modules and the Internet service monitors.

Operation and configuration

The Databridge and its component modules are configured through properties files. The properties determine the operation of the Databridge and its component modules.

The component modules are:

- IBM Tivoli Monitoring module, which sends test results to IBM Tivoli Monitoring for reporting in workspaces.
- The ObjectServer module, which sends events to a IBM Tivoli Netcool/OMNIbus ObjectServer.
- Datalog module, which generates XML datalog files for archiving or simple, external reporting purposes. Use this module only with IBM Tivoli Netcool/OMNIbus.

Figure 9 on page 87 presents an overview of the Databridge, including its component modules, properties files, and the systems to which the modules connect. See each module's administration section for details about properties and connections.

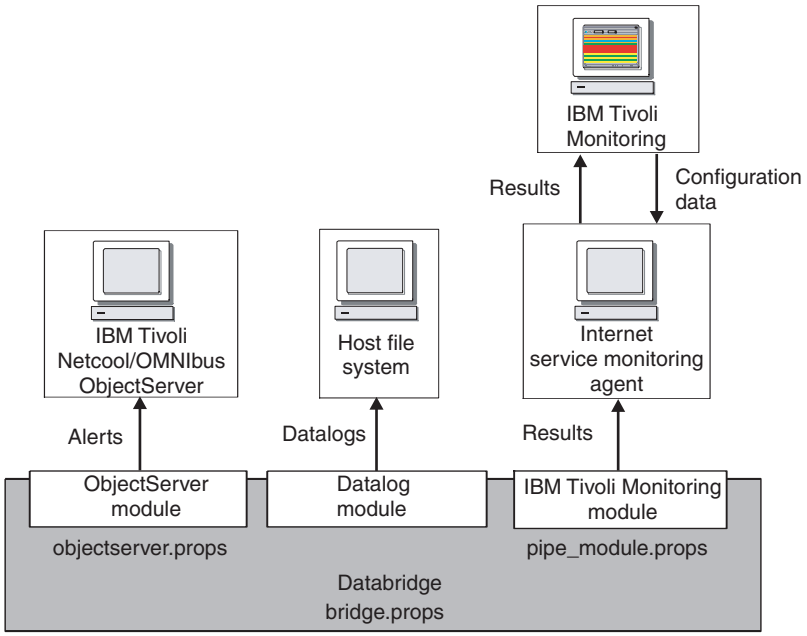


Figure 9. Databridge overview

Configuring the Databridge

The Databridge must be configured to receive data from the Internet service monitors and to forward that data to its component modules for further processing.

Table 12 lists the files associated with the Databridge. The **Properties file**, **Store And Forward file**, and **Log file** are described in more detail in the appropriate sections.

Table 12. Databridge files and their location

Databridge file	**Location and/or name**
Executable file	`$ISMHOME/platform/arch/bin/nco_m_bridge`
Properties file	`$ISMHOME/etc/props/bridge.props`
Store and Forward file	Name and location are specified by properties in the `bridge.props` file. The default name and location is `$ISMHOME/var/sm_bridge.saf`
Log file	`$ISMHOME/log/bridge.log`
Error log file	`$ISMHOME/log/bridge.err`

Databridge properties and command line options

The properties file controls the operation of the Databridge. The properties specify the modules to which information is forwarded for further processing, the time at which log files should be generated, and any other Databridge properties.

You can modify the property values in the Databridge properties file or specify the values on the command line. If you make any changes to the properties file, you must restart the Databridge before the changes take affect.

By default, the Databridge uses the properties file `$ISMHOME/etc/props/bridge.props`, however you can instruct it to use an alternative file by starting it using the command:

```
$ISMHOME/platform/arch/bin/nco_m_bridge -propsfile file
```

where *arch* is the specific platform name and *file* is the path and filename of the required properties file.

Table 13 lists the properties and command-line options available for the Databridge. You can also obtain a list of the available command-line options using the command:

```
$ISMHOME/platform/arch/bin/nco_m_bridge -help
```

Table 13. Databridge properties and command-line options

Property name	Parameter	Command line option	Description
MaxCCA	integer	-maxcca	Sets the maximum number of concurrent connections supported by the Databridge at any one time. **Note:** Setting MaxCCA to a high value may severely affect the performance of the Databridge. Default: 30
MessageLog	string	-messagelog	Specifies the location of the Databridge log file. Default: $ISMHOME/log/bridge.log
Module*n*PropFile	string	Not applicable	The name of the modules properties file.
Module*n*SharedLib	string	Not applicable	The name of the modules shared library file.
MsgDailyLog	0|1	-msgdailylog	Enables generation of a daily log file: • 0 - Enabled • 1 - Disabled
MsgTimeLog	string	-msgtimelog	Specifies the time (in 24-hour format HHMM) after which the Databridge generates a daily log if MsgDailyLog has the value True. Default: 0000 - 12:00am

Table 13. Databridge properties and command-line options (continued)

Property name	Parameter	Command line option	Description
NoRecover	0|1	-norecover	Suppresses automatic recovery of the store and forward file. • 0 - Disabled • 1 - Enabled
Not applicable	string	-propsfile	Specifies the path and filename of the Databridge properties file. Default: $ISMHOME/etc/props/bridge.props
QFile	string	-qfile	Sets the name of the store and forward file. Default: $ISMHOME/var/sm_bridge.saf
QSize	integer	-qsize	Sets the reserved size of the store and forward file (in bytes). Default: 51200000
SocketBacklog	integer	-socketbacklog	Sets the maximum number of pending connections in the Databridge socket's listen queue. If the length of the queue exceeds this value, the Databridge refuses further connection requests. Default is 10.
SocketBufferSize	integer	-socketbuffersize	Sets the buffer size of the Databridge socket connection (in kilobytes). Specify a minimum of 8. Default: 256
SocketPort	integer	-socketport	Specifies the port number on which the Databridge listens for connections. Default: 9510
SocketTimeout	integer	-sockettimeout	Sets the timeout of Databridge socket connections (in seconds). Default: 10
BridgeSSL AuthenticatePeer	0|1	-bridgessl authenticatepeer	Specifies whether the Databridge needs to authenticate monitor encryption certificates. • 0 - Disabled • 1 - Enabled
BridgeSSL CertificateFile	string	-bridgessl certificatefile	Specifies the path and filename of the digital Bridge SSL certificate. Default: $ISMHOME/certificates/bridgeCert.pem

Table 13. Databridge properties and command-line options (continued)

Property name	Parameter	Command line option	Description
`BridgeSSL Encryption`	`0|1`	`-bridgessl encryption`	Specifies whether Bridge SSL encryption applies. • 0 - Enabled • 1 - Disabled **Note:** Set to the same value as for the monitors.
`BridgeSSL KeyFile`	string	`-bridgessl keyfile`	The path and the filename of the Bridge SSL private key file. Default: `$ISMHOME/certificates/bridgeKey.pem`
`BridgeSSL KeyPassword`	string	`-bridgessl keypassword`	The password used to encrypt the Bridge SSL private key. Default: `tivoli`
`BridgeSSL Truststore`	string	`-bridgessl truststore`	The path and file name of the Trusted certificate file for authentication. This is only required when using the AuthenticatePeer setting. Default: `$ISMHOME/certificates/trust.pem`
On Windows platforms, specify path separators using \\ in place of \.			

Store And Forward file

If the Databridge is unable to forward data to IBM Tivoli Netcool/OMNIbus, it stores all of the data it would normally send in a Store And Forward (SAF) file. When IBM Tivoli Netcool/OMNIbus becomes available again, it processes all of the events stored in the SAF file.

The `QFile` and `QSize` properties in the Databridge properties file determine the name, location and operation of the store and forward processing.

Log file

The Databridge sends daily messages about its operations to a message log file.

By default, the name of this file is `$ISMHOME/log/bridge.log`. It is updated at midnight (12:00am). The Databridge properties `MsgDailyLog` and `MsgTimeLog` control the operation of message logging.

Starting the Databridge

To start the Databridge, use either the Windows Services console, or the command-line or shell prompt.

About this task

Note: If the ObjectServer module is connected to the Databridge, ensure that its target system is running before starting the Databridge. If any of the Databridge modules fails to initialize correctly, the Databridge will not start.

To start the Databridge from the command prompt on Windows, use the command:

```
%ISMHOME%\platform\win32\bin
```

To start the Databridge from the command-line or shell prompt on UNIX, use the command:

```
$ISMHOME/bin/nco_m_bridge
```

Procedure

To start the Databridge from the Services console:

1. From the Windows desktop, select **Start** > **Administrative Tools** > **Services**.
2. From the list of services, select the service named `NCO BRIDGE Internet Service Monitor`, then select **Start** from the context menu.

Connecting modules

The Databridge properties file defines the modules to connect to the Databridge.

About this task

Each `Module`*`n`*`SharedLib` and `Module`*`n`*`PropFile` property pair defines the connection for one module. Modules are loaded in order of definition, starting from `Module0`.

To connect individual modules to the Databridge:

1. In the Databridge properties file, identify the next available `Module`*`n`*`SharedLib` and `Module`*`n`*`PropFile` property pair.
2. Set `Module`*`n`*`SharedLib` to the name of the module's shared library (its binary implementation).
3. Set `Module`*`n`*`PropFile` to the full path of the module's properties file.

To connect all modules to the Databridge on UNIX, add the following entries to the Databridge properties file `$ISMHOME/etc/props/bridge.props`:

```
[1]Module0PropFile        : "$ISMHOME/etc/props/objectserver.props"
[2]Module0SharedLib       : "libSMModuleObjectServer"
[3]Module1PropFile        : ""
[4]Module1SharedLib       : "libSMModuleDatalog"
[5]Module2PropFile        : "$ISMHOME/etc/props/pipe_module.props"
[6]Module2SharedLib       : "libSMModulePipe"
```

In this example, lines 1 and 2 connect the ObjectServer module, lines 3 and 4 connect the Datalog module, lines 5 and 6 connect the IBM Tivoli Monitoring (`pipe`) module. The Datalog module does not have a properties file, so the entry for the properties file has the value `""`.

Disabling modules:

To disable a module, set the corresponding Module*n*SharedLib property to "NONE" and the Module*n*PropFile property to "". All other modules that have a value higher than *n* are also ignored.

Connecting monitors

Internet service monitors connect to the Databridge over TCP. Each monitor has a set of properties that configure the connection to the Databridge.

About this task

To connect a monitor to the Databridge, set the value of the BridgePort property defined in the monitor's properties file to the value of the SocketPort property defined in the Databridge properties file. The default value of each monitor's BridgePort property and the Databridge's SocketPort property is 9510.

The Databridge supports SSL encryption of the test results that it receives from the monitors. To encrypt a monitor's test results, set the values of the BridgeSSL properties defined in the monitor's properties file to the values of the BridgeSSL properties defined in the Databridge properties file. To encrypt all monitors' test results, all monitors must have the same BridgeSSL properties.

Configuring the IBM Tivoli Monitoring module

The IBM Tivoli Monitoring module directs test results to the Internet service monitoring agent. The monitoring agent converts this data to the required format and distributes it to the Tivoli Enterprise Monitoring Server.

You configure both the IBM Tivoli Monitoring module and the Internet service monitoring agent through their respective properties files.

IBM Tivoli Monitoring properties

You configure the operation of the IBM Tivoli Monitoring module by modifying the property values defined in the module properties file.

The module properties file is named pipe_module.props. This file is located in the $ISMHOME/etc/props/ directory.

Table 14 lists the properties available for the module. If you make changes to the properties, you must restart the Databridge for those changes to take effect.

Table 14. IBM Tivoli Monitoring module properties

Property name	Type	Description
TEMAHOST	string	The name of the host running the monitoring agent. Default: localhost
TEMAPORT	integer	The port number used by the host. Default: 9520

Internet Service Monitoring agent properties

You configure the operation of the Internet service monitoring agent by modifying the property values defined in the monitoring agent properties file.

The monitoring agent properties file is named `kisagent.props`. This file is located in the `$ISMHOME/etc/props/` directory.

Table 15 lists the properties available for the monitoring agent.

Table 15. Monitoring agent properties

Property name	Type	Description
`TEMAPORT`	integer	The port number used by the host. This must be the same as the port number for the `TEMAPORT` property listed in the module properties file. Default: `9520`
`ObsoleteDuration`	integer	The time, in seconds, after which any data that has not been updated is deleted from the monitoring agent's memory. Data might not be updated when, for example, a profile element has been stopped or a network failure has occurred. **Note:** Do not set the `ObsoleteDuration` time to a value less than the poll interval because this results in loss of data between poll intervals. Default: `900`
`AggDuration`	integer	The time, in seconds, after which the monitoring agent stops data from being aggregated and reported in statistical workspaces. Any data that is older than the specified time is deleted from the monitoring agent's memory. Older data is calculated by comparing the interval between the start and the current time to the aggregate duration time. If the interval is greater than the aggregate duration time, 10% of the older data is removed and the start time is increased by 1/10th of the interval. The monitoring agent performs this calculation every five minutes. Default: `3600`
`ManageServices`	`0`|`1`	Starts and stops all monitors and the Databridge when the Internet Service Monitoring agent is started or stopped. `1` is enabled, and `0` is disabled. Default: `1`

Connecting the Internet Service Monitoring agent

The connection between the Internet service monitoring agent and the IBM Tivoli Monitoring module is created when you install Internet Service Monitoring.

Connection information for UNIX is located in `$CANDLE_HOME/config/`. For Windows, connection information is stored in the `kisenv` configuration file located in `%CANDLE_HOME%/TMAITM6/`.

Configuring the ObjectServer module

The ObjectServer module converts events into alerts which it then forwards to the IBM Tivoli Netcool/OMNIbus ObjectServer.

The operation of the ObjectServer module is very similar to IBM Tivoli Netcool/OMNIbus probes except that its data source is the event stream received from monitors rather than a network device. Using a rules file, the ObjectServer module converts elements contained in monitor events into fields in IBM TivoliNetcool/OMNIbus alerts, which it sends to an ObjectServer. The ObjectServer module uses *all* rules in the rules file. Table 16 lists the files associated with the ObjectServer module.

Table 16. ObjectServer module files and their location

Module file	Location and/or name
Library	`libSMModuleObjectServer`
Log file	`$ISMHOME/log/objectserver.log`
Properties file	`$ISMHOME/etc/props/objectserver.props`
Rules file	`$ISMHOME/etc/rules/objectserver.rules`

Releases before Internet Service Monitoring 6.0.0 included multiple ObjectServer modules, however version 6.0.0 and higher provides one common module, `libSMModuleObjectServer`, for connecting to all ObjectServers.

ObjectServer properties

You can configure the operation of the ObjectServer module by modifying the property values defined in the ObjectServer module properties file.

Table 17 lists the properties available for the ObjectServer module. If you make changes to the properties, you must restart the Databridge for those changes to take effect.

Table 17. ObjectServer module properties

Property name	Type	Description
`AuthPassword`	string	Specifies the password associated with the username that the ObjectServer module uses to authenticate itself when the target ObjectServer is running in secure mode. This password must be encrypted with the `nco_crypt` utility. Default: `""`
`AuthUserName`	string	Specifies the username that the ObjectServer module uses to authenticate itself when the target ObjectServer is running in secure mode. Default: `""`

Table 17. ObjectServer module properties (continued)

Property name	Type	Description
AutoSAF	0|1	Enables automatic store and forward mode: • 0 - Disabled • 1 - Enabled
Buffering	0|1	Enables buffering of alerts: • 0 - Disabled • 1 - Enabled
BufferSize	integer	Sets the number of status messages that the ObjectServer module stores in its buffer. Default: 10
LookupTableMode	0|1|3	Specifies how table lookups are performed: • 1 - All external lookup tables are assumed to have a single value column. Tabs are not used as column delimiters. • 2 - All external lookup tables are assumed to have multiple columns. If the number of columns on each line is not the same, an error is generated that includes the file name and the line on which the error occurred. • 3 - The rules engine attempts to determine the number of columns in the external lookup table. An error is generated for each line that has a different column count from the previous line. The error includes the file name and the line on which the error occurred. For detailed information about lookup table operations, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*.
MaxLogFileSize	integer	Specifies the maximum size of the log file (in bytes) before it rolls over to the objectserver_old.log file. Default: 1
MaxRawFileSize	integer	Specifies the maximum size of the raw capture file in kilobytes. If RawCaptureFileBackup is disabled, this property is ignored. If RawCaptureFileBackup is enabled, this property specifies the approximate file size at which a new file is started. Default: -1 (unlimited size)
MaxSAFFileSize	integer	Specifies the maximum size of the store and forward file (in bytes). Default: 1MB
MessageLog	string	Specifies the location of the ObjectServer module log file. Default: $ISMHOME/log/objectserver.log

Table 17. ObjectServer module properties (continued)

Property name	Type	Description
NetworkTimeout	integer	Specifies the period (in seconds) after which the connection to an ObjectServer will time out if a network failure occurs. Default: 0 (no timeout)
RawCapture	0|1	Controls the raw capture mode: • 0 - Disabled • 1 - Enabled
RawCaptureFileAppend	0|1	If specified, new data is appended to the existing raw capture file, instead of overwriting it: • 0 - Overwrite • 1 - Append
RawCaptureFileBackup	0|1	Enables backup of the raw capture file when it exceeds the maximum file size specified by the MaxRawFileSize property: • 0 - Disabled • 1 - Enabled
RawCaptureFile	string	The name of the raw capture file. Default: $ISMHOME/var/objectserver.cap
RetryConnectionCount	integer	The number of events the probe must process in store and forward mode before trying to connect to the ObjectServer. Default: 15
RetryConnectionTimeOut	integer	The number of seconds the probe must process events in store and forward mode before trying to connect to the ObjectServer. Default: 30
RulesFile	string	Specifies the location and filename of the ObjectServer module rules file. Default: $ISMHOME/etc/rules/objectserver.rules
SAFFileName	string	The name of the ObjectServer module store and forward file. Default: $ISMHOME/var/objectserver.saf.*name*, where *name* is the name of the target ObjectServer.
Server	string	Specifies the name of the ObjectServer to which the ObjectServer module sends the events generated from the Internet service monitors. Default: NCOMS
ServerBackup	string	Defines a secondary ObjectServer if the primary ObjectServer connection fails. If NetworkTimeout is set and a virtual ObjectServer is in use, use ServerBackup to refer to your secondary ObjectServer. Default: ""

Table 17. ObjectServer module properties (continued)

Property name	Type	Description
StoreAndForward	0|1	Controls the store and forward operations: • 0 - Disabled • 1 - Enabled
On Windows, specify path separators using \\ in place of \.		

Raw capture mode:

Like probes, the ObjectServer module provides a raw capture mode in which it saves the complete stream of events into a file without any processing by the rules file. This mode is useful for auditing, recording, or debugging the ObjectServer module.

The RawCapture, RawCaptureFile, RawCaptureFileAppend, RawCaptureFileBackup, and MaxRawFileSize ObjectServer properties control the operation of raw capture mode.

Store And Forward mode:

The ObjectServer module provides a Store And Forward (SAF) mode for fault-tolerant operation.

The module enters SAF mode if it is unable to send alerts to the target ObjectServer, for example if the network connection between the two systems fails. In SAF mode, the module stores alert information in a file instead of sending it to the ObjectServer. When the ObjectServer becomes available again, the module forwards all alerts stored in the SAF file and returns to normal operation.

The AutoSAF and StoreAndForward properties control the operation of SAF mode. When AutoSAF has the value 1, the module enters SAF mode if the target ObjectServer is either unavailable when the Databridge starts, or if it becomes unavailable while the Databridge is running. When StoreAndForward has the value 1, the module only enters SAF mode if the target ObjectServer becomes unavailable while the Databridge is running; the Databridge will not start if the ObjectServer is unavailable.

If both AutoSAF and StoreAndForward have the value 1, AutoSAF takes precedence. The SAFFileName and MaxSAFFileSize ObjectServer properties define the name and size of the SAF file.

Rules files

The ObjectServer module and each Internet service monitor provide a rules file for converting monitor events into alert information which is used by IBM Tivoli Netcool/OMNIbus.

The ObjectServer module's rules file acts as a wrapper for individual Internet service monitor rules files, each of which contains rules specific to the elements generated by that monitor.

Note: To define lookup tables for processing events generated by a specific monitor, define them at the start of the ObjectServer rules file rather than in the monitor rules file. Rules file parsing requires that lookup tables appear before any

processing statement, so any lookup tables that are defined in a monitor rules file result in parser errors in the ObjectServer rules file.

If you modify the rules files, you must restart the Databridge for the changes to take effect. On UNIX, you can force the ObjectServer module to re-read the rules files by issuing the command `kill -HUP pid`, where `pid` is the process ID of the ObjectServer module process. For information about rules files and their syntax, see the *Netcool/OMNIbus Probe and Gateway Guide*. By default, rules files are located in the `$ISMHOME/etc/rules` directory.

Log file

The ObjectServer module reports information about its operations to a log file.

The `MessageLog` property specifies the location of the log file. The `MaxLogFileSize` property specifies the size of the log file before it rolls over to the `objectserver_old.log` file.

Connecting to the ObjectServer

The ObjectServer module properties file defines the IBM Tivoli Netcool/OMNIbus ObjectServer to which the ObjectServer module sends alert information.

To configure the module to connect to the ObjectServer, set the `Server` property to the name of the ObjectServer.

Note: When connecting the ObjectServer module, ensure that the target IBM Tivoli Netcool/OMNIbus is running before starting the Databridge.

If the target ObjectServer is running in secure mode, the ObjectServer module must authenticate itself when connecting to it. The `AuthUserName` and `AuthPassword` properties define the credentials that the ObjectServer module uses during authentication. Encrypt passwords using the `nco_g_crypt` utility. For more information about this utility, see the *IBM Tivoli Netcool/OMNIbus ObjectServer Administration Guide*.

Connecting to the ObjectServer on UNIX

To configure the connection of the ObjectServer module to an ObjectServer on UNIX:

- Ensure that the connections data file `$ISMHOME/objectserver/etc/omni.dat` contains an entry for the target ObjectServer. If it does not, add one to the file using the format:

  ```
  [OBJSERVERNAME] {Primary: hostname port}
  ```

  where `OBJSERVERNAME` is the name of the target ObjectServer, and `hostname` and `port` represent the DNS name, or IP address, and port of the system on which the ObjectServer is running.
- Generate an interfaces file using the command:

  ```
  $ISMHOME/objectserver/bin/nco_igen
  ```

Connecting to the ObjectServer on Windows

To configure the connection to the ObjectServer on Windows, add the following entry for the target ObjectServer in the connections data file `$ISMHOME\objectserver\ini\sql.ini`:

```
[OBJSERVERNAME]
master=NLWNSCK,hostname,port
query=NLWNSCK,host,port
```

where `OBJSERVERNAME` is the name of the target ObjectServer, and `hostname` and `port` represent the DNS name, or IP address, and port of the system on which the ObjectServer is running.

Configuring the Datalog module

The Datalog module generates XML datalog files from the test results received from the Internet service monitors. You can store the datalog files on a local host for archiving purposes or for generating simple reports.

The Datalog module does not have a properties file. The file associated with the Datalog module is the `libSMModuleDatalog` library file.

Note: Datalogs can occupy a large amount of disk space. See "Disk space requirements" on page 39 for a guideline on calculating the amount of disk space required.

Datalog files

Datalog files store the monitor data generated for a profile element over a 24-hour period. Datalog files can be used for testing or basic reporting purposes and are mainly used by customers who have developed their own reporting tools.

Datalogs are located in subdirectories in `$ISMHOME/datalogs` on the local host. The datalogs are grouped by profile and profile element name. Datalog filenames have the format *YYYYMMDD*`.xml`, where *YYYYMMDD* represents the monitoring period.

The format of the monitor data in the datalog files is defined by a set of default datalogs.

Default datalog file

The default datalog file determines the format of the monitor data in datalog files.

Each time the Datalog module generates a new datalog file, it uses the default datalog file for the corresponding monitor as a template. The default datalog files are stored in `$ISMHOME/datalogs/default`.

Note: Before modifying a default datalog file, create a backup copy of the original.

Enabling data logging

Datalog files store the test results data received from the Internet services monitors. When you enable data logging, you can store the datalog files for archiving purposes or for generating simple reports.

Procedure

To enable data logging:

1. Make sure that the Datalog module is connected to the Databridge.
2. Set the `Datalog` property in each monitor's properties file to `1`.

Disk space requirements

Datalogs are created by the Databridge Datalog module. These datalogs can occupy a large amount of disk space.

Note: Typically the Datalog module is not required; it is disabled by default.
To calculate the amount of disk space required by datalog files, use the following formula:

```
(600 / poll_interval) × 15 KB per profile element per day
```

For example, a single HTTP profile element polling a web page every 5 minutes for one year would require 10.95 MB of disk space:

```
(600 / (5 × 60)) × 15 × 365 = 10.95 MB
```

Troubleshooting the Databridge

Use the following guidelines when troubleshooting problems with the Databridge:

- Check the Databridge error log, `$ISMHOME/log/bridge.err`, for messages about errors occurring when the Databridge starts:
  - If the Databridge log file contains the following error:

    `Failed to open Properties file: filename`

    check that the path and filename of the properties file specified in the Databridge properties file are correct. On Windows, specify path separators using \\ in place of \.
  - If the log file indicates that the Databridge cannot load a module, check for errors in the module's shared library name in the Databridge properties file.
- If the Databridge log file contains the following, IBM Tivoli Netcool/OMNIbus related, error:

  `Failed to read rules - aborting`
  - check that the target IBM Tivoli Netcool/OMNIbus ObjectServer is running.
  - confirm that property values in the ObjectServer module properties file are correct.
  - confirm that the rules files are correct.
- If you have connected the ObjectServer module to the Databridge and the Databridge will not start, confirm that the target IBM Tivoli Netcool/OMNIbus is running before you start the Databridge, or set the ObjectServer module `AutoSAF` property to `1`.

Chapter 6. Installing Response Time

Administrators can install Response Time monitoring agents and related software.

Installing Response Time monitoring agents and related software

You install the monitoring agent and add the application support to display monitoring data in the Tivoli Enterprise Portal.

This chapter tells how to install and configure a Tivoli Enterprise Monitoring Agent. You can install a monitoring agent *either* on a computer by itself *or* on a computer with an installed portal server, monitoring server, and/or portal.

Note: If you install the monitoring agent on a computer where the portal server, monitoring server, and portal are already installed, the installation asks if you want to install application support, depending on what is installed on the monitoring agent computer; the procedure for installing application support is essentially the same as installing a monitoring agent.

Table 18. Installation checklist

What to do	Where to find more information
Obtain the installation software.	Either download the software from Passport Advantage (see the Download information on the ITCAM for Transactions Information Center) or use a product DVD. Talk to your Tivoli System Administrator.
Verify the software and hardware requirements for the agent you want to install have been met.	Prerequisites
Collect the necessary information for the installation and configuration.	Information to collect before you begin installation and configuration
Install Rational products: • Rational Performance Tester *(Required if you want to record RPT scripts to playback on Robotic Response Time)* • Rational Robot *(Optional)*	"Installing integration support for Rational Performance Tester" on page 130 "Installing Rational Robot" on page 135
Install the Tivoli Enterprise Monitoring Agent.	• "Install a monitoring agent on Windows systems" on page 102 • "Installing a monitoring agent on Linux or UNIX computers" on page 116 • "Performing a silent installation" on page 126 **Note:** When specifying the path where you want to install an agent, do not include blank spaces in the path name.
Install Tivoli Enterprise Monitoring Agent specific application support.	• "Install application support for Windows systems" on page 107 • "Install application support for Linux and UNIX" on page 120

Table 18. Installation checklist (continued)

What to do	Where to find more information
Enable file transfer (UNIX and Linux)	"Enabling File Transfer (UNIX and Linux)" on page 122
Configure the monitoring agent.	• "Configuring Application Management Console" on page 143 • "Configuring Client Response Time" on page 149 • "Configuring Robotic Response Time" on page 154 • "Configuring Web Response Time" on page 164
Verify the installation of a Tivoli Enterprise Monitoring Agent.	"Verify installations of Response Time monitoring agents" on page 125
Configure the Eclipse help server.	Chapter 12, "Configuring the Eclipse help server," on page 321
Set up remote deployment. *(Optional)* Response Time supports remotely deploying the monitoring agent across your environment from a central location, the Tivoli Enterprise Monitoring Server.	Chapter 11, "Working remotely," on page 309
Configure IBM Tivoli Monitoring to forward events to Tivoli Enterprise Console. *(Optional)*	Appendix E, "Tivoli Enterprise Console event mapping," on page 349
Start the monitoring agent.	"Starting and stopping monitoring agents" on page 326
Configure for historical data collection. *(Optional)*	See the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide*.
Install a language pack. *(Optional)* You can install local language support on each computer on which the Tivoli Enterprise Portal is located	Appendix A, "Installing and uninstalling the language pack," on page 329

Install a monitoring agent on Windows systems

Procedure

1. Log on to the Windows system using a user ID with the Administrator authority.
2. Obtain the installation software by downloading it or inserting the product CD.
3. Start the installation wizard by double-clicking the `setup.exe` file in the `\WINDOWS` subdirectory.

   Tip: On Windows Server 2008 systems, if instead of the installer you see the following message, right-click the `setup.exe` file in the file explorer and select **Run as Administrator**.

   ```
   Your logon ID must have Administrator rights to install IBM Tivoli Composite
    Application Manager for Transactions
   ```

4. In the Welcome window, click **Next**. The Install Prerequisites window is displayed.

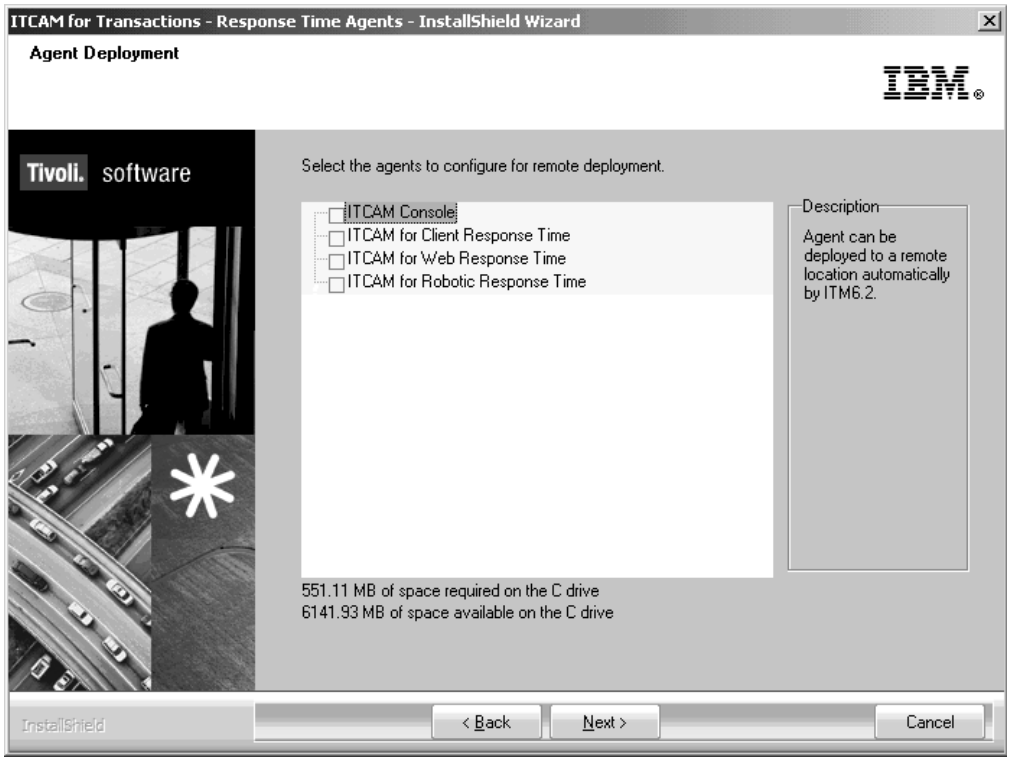
5. In the Install Prerequisites window, follow the instructions and select the appropriate check boxes for the required versions of IBM Global Security ToolKit (GSKit) 7.0.3.18 or higher and IBM Java version 1.5.0 IBM Windows 32 build pwi32devifx-20070706 (SR5 + IZ00983). In the **Choose installation drive for both** field, enter the drive on which to install the software, and then click **Next** to proceed with the installation process. Once installation of the prerequisite software is complete, you will be automatically returned to the IBM Tivoli Composite Application Manager for Transactions InstallShield Wizard.
6. The Software License Agreement window is displayed. If you accept the terms of the license agreement, click **Accept** to continue, otherwise click **Decline** to stop the installation process. You must accept the terms of the license agreement to continue the installation process.
7. Choose the directory where you want to install the product. The default directory is `C:\IBM\ITM`. Click **Next** to display the User Data Encryption Key window.

   Note: If there is an IBM Tivoli Monitoring monitoring agent on the computer, you cannot change the installation directory.
8. Type a 32-character encryption key and click **Next**. You can use the default key. Do not use the = or ' or | characters in your key.
9. Click **Next** to display the Select Features window.

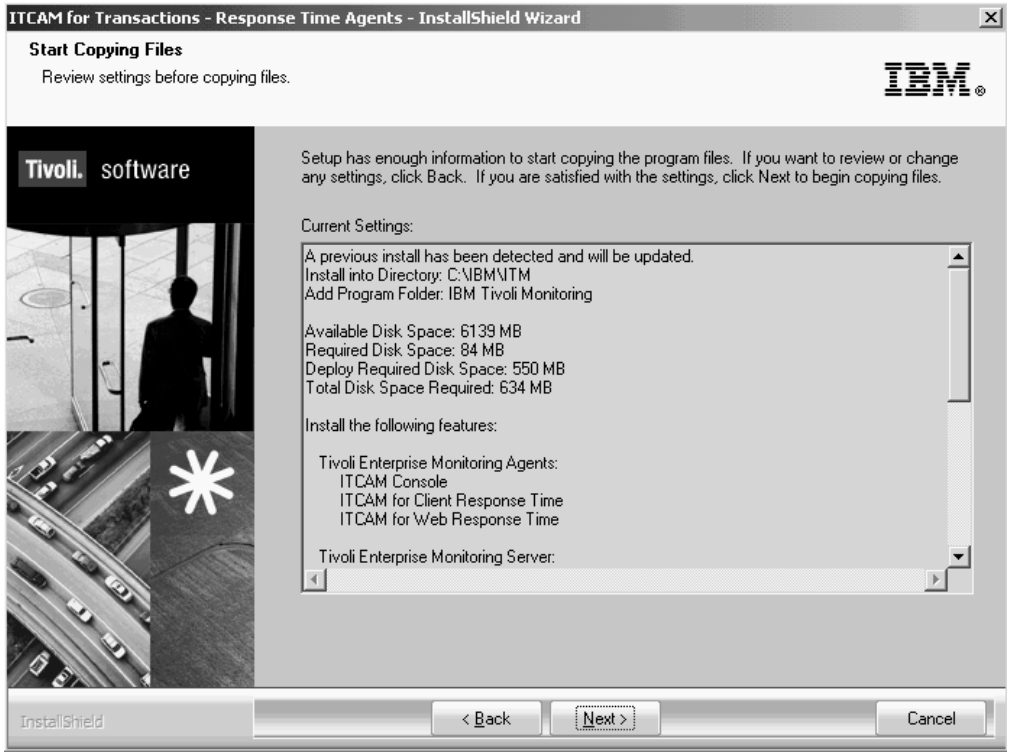
10. Expand **Tivoli Enterprise Monitoring Agents**, and select the agents that you want to install from the provided list by selecting the check box next to the agent name. Leave the **Tivoli Enterprise Monitoring Agent Framework** check box selected.

    Remember: Depending on your operating system, some agents are not supported for installation. Before selecting one or more agents to install, ensure that they are supported on your operating system by checking the Prerequisites pages in the ITCAM for Transactions Information Center.

11. Click **Next** to display the Agent Deployment window. If you are installing locally, do not select any agents. If you are deploying an agent to a remote server, select the agent to enable remote deployment. See Chapter 11, “Working remotely,” on page 309 for further information.

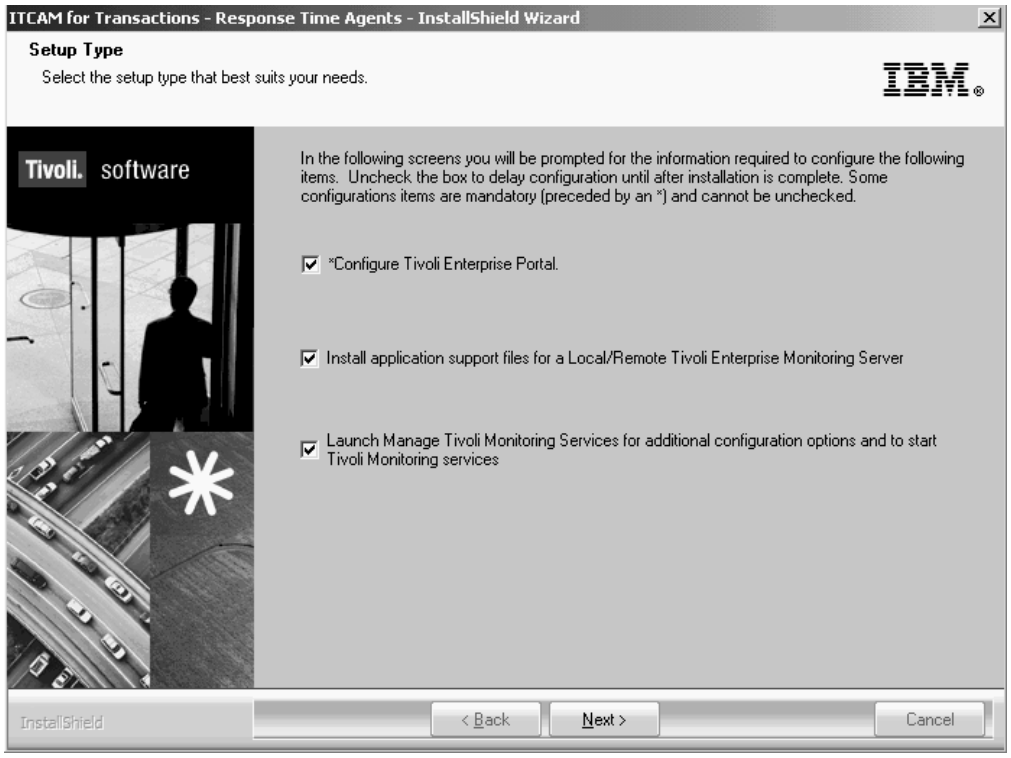


12. Click **Next** to display the Response Time Information window.
13. Click **Next** to display the Start Copying Files window.



14. Click **Next** to continue, or click **Back** to modify your selection.

15. A message is displayed stating that you will not be able to cancel the installation or upgrade after this point. Click **Yes** to continue. Files will now be copied to your computer. This might take several minutes. A Setup Status window provides status messages about the installation progress.
16. Before the Setup is complete, a Setup Type window is displayed. To configure your Tivoli Enterprise Monitoring agent and complete the installation, select the items that you want to configure and click **Next**. Otherwise, clear the check boxes and continue the configuration at another time. You cannot clear the check box that is preceded by an asterisk.



17. Go to the "Follow-up or related tasks" section. Choose the Tivoli Enterprise Monitoring agent that you want to configure and continue the installation or configuration.

Follow-up or related tasks

Make sure to perform the following follow-up tasks:

- Configure the monitoring agent that you just installed. It is recommended to complete the configuration during the installation.
  - "Configuring Application Management Console (Windows)" on page 143
  - "Configuring Client Response Time (Windows)" on page 150
  - "Configuring Web Response Time (Windows)" on page 165
  - "Configuring Robotic Response Time (Windows)" on page 154
- "Verify installations of Response Time monitoring agents" on page 125
- Chapter 12, "Configuring the Eclipse help server," on page 321

Install application support for Windows systems

For each monitoring agent installed, you must install agent-specific application support on IBM Tivoli Monitoring components. You can choose to install application support on all components at one time, or you can install the application support on components separately. If installing components separately, the following order is recommended:

1. Tivoli Enterprise Monitoring Server
2. Tivoli Enterprise Portal Server
3. Tivoli Enterprise Portal Desktop Client

The procedure for installing agent-specific application support on the components is nearly identical. The following steps provide the basic procedure and note the differences in Table 19 on page 108. Repeat the procedure for each of the components on which you must install application support.

Note: Before installing application support, stop the Tivoli Enterprise Monitoring Server. **If you do not stop the server manually, the installation software stops it automatically during the installation process**. You should warn other users before beginning the installation.

Table 19. Application support installation summary by component type

For Tivoli Enterprise Monitoring Server application support	For Tivoli Enterprise Portal server and desktop client
1. In the Select Features window, select **Tivoli Enterprise Monitoring Server**. 2. Click **Next** to display the Start Copying Files window. 3. Review the installation summary details and click **Next** to start the installation. 4. A message is displayed. Read the instructions and click **Yes** to continue. 5. In the Setup Type window, select the setup type. Default setup and configuration options are selected. It is recommended you set up and configure the Tivoli Enterprise Monitoring Server now, but you can clear the check boxes to delay setup and configuration until after the installation is complete. 6. Click **Next** to display the Tivoli Enterprise Monitoring Server Configuration window. This window displays the default responses based on the Tivoli Enterprise Monitoring Server configuration setup for IBM Tivoli Monitoring. 7. Click **OK** to display the Hub TEMS Configuration window. 8. Define the default values the agents use to connect to the monitoring server. 9. Click **OK** to display the Add Application Support to the TEMS window. 10. Specify if the location of the Tivoli Enterprise Monitoring Server is on this computer or remote. 11. Click **OK** to display the Select the Application Support to Add to TEMS window. 12. Select each agent for which you want to install application support. Clear each agent for which application support is already installed. Click **OK** to display the Application Support Addition Complete window. 13. Click **Next** to display the Configuration Defaults for Connecting to a TEMS window. 14. Specify the default values for the monitoring agent to use when communicating with the monitoring server. 15. Click **OK** to display the Installation Wizard Complete window. 16. Click **Finish** to complete the installation.	1. In the Select Features window, select **Tivoli Enterprise Portal Server** or **Tivoli Enterprise Portal Desktop Client**, and click **Next**. 2. The Agent Deployment window is displayed. Make sure that no agent is selected. Click **Next**. 3. Click **Next** to display the Response Time Information window. 4. Click **Next** to display the Start Copying Files window. 5. Review the installation summary details and click **Next** to start the installation. 6. A message is displayed stating that you will not be able to cancel the installation or upgrade after this point. Click **Yes** to continue. 7. In the Setup Type window, select **Configure Tivoli Enterprise Portal** and click **Next**. 8. In the TEPS Hostname window, type the host name of the system where the Tivoli Enterprise Portal Server is installed and click **Next**. 9. Click **Finish** to complete the installation. If you are installing application support files on Tivoli Enterprise Portal Server, the server is restarted automatically.

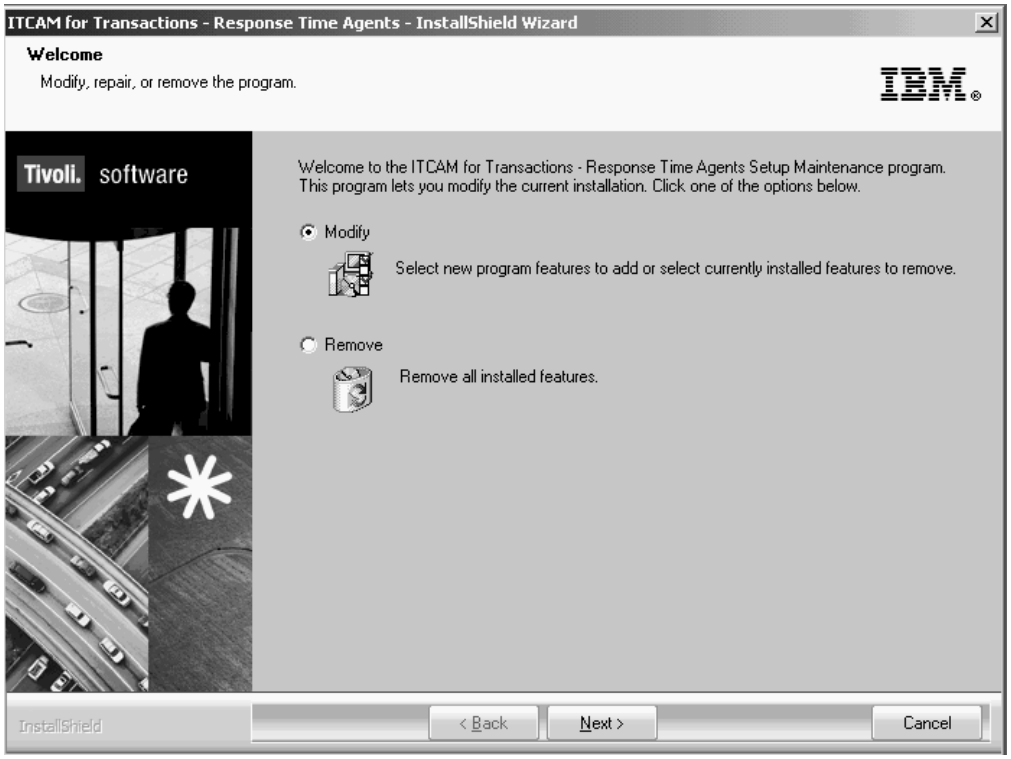
1. For each component for which you are installing application support, the installation automatically stops the component, or you can manually stop each component. For more information on manually stopping a component, see:
   - Stop the Tivoli Enterprise Monitoring Server.
   - Stop the Tivoli Enterprise Portal server.
   - Stop the Tivoli Enterprise Portal desktop client.
2. Start the installation wizard by double-clicking the `setup.exe` file in the `\WINDOWS` subdirectory.

Tip: On Windows Server 2008 systems, if instead of the installer you see the following message, right-click the `setup.exe` file in the file explorer and select **Run as Administrator**.

```
Your logon ID must have Administrator rights to install IBM Tivoli Composite
 Application Manager for Transactions
```

3. Select **Modify**, click **Next**, and go to step 7.

   Note: If you are installing the support separately from the agent and you have already installed an agent on this computer, a window similar to the following window.



4. In the Welcome window, click **Next**. The Install Prerequisites window is displayed.

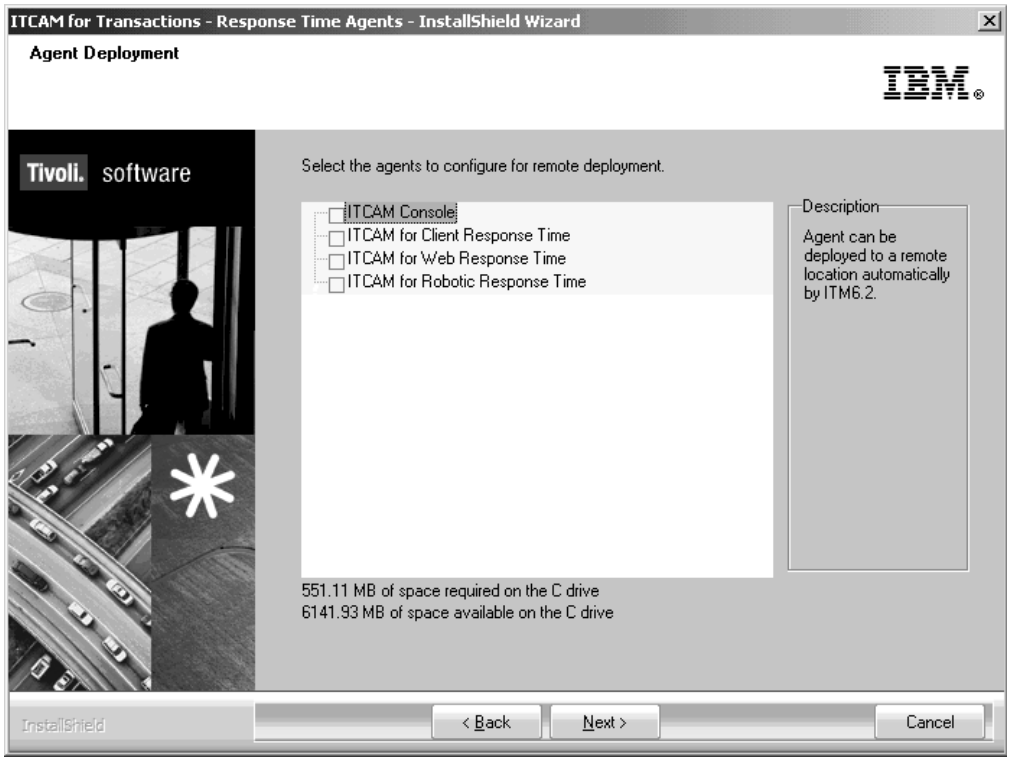
5. In the Install Prerequisites window, follow the instructions and select the appropriate check boxes for the required versions of IBM Global Security ToolKit (GSKit) 7.0.3.18 or higher and IBM Java version 1.5.0 IBM Windows 32 build pwi32devifx-20070706 (SR5 + IZ00983). In the **Choose installation drive for both** field, enter the drive on which to install the software, and then click **Next** to proceed with the installation process. Once installation of the prerequisite software is complete, you will be automatically returned to the IBM Tivoli Composite Application Manager for Transactions InstallShield Wizard.
6. The Software License Agreement window is displayed. If you accept the terms of the license agreement, click **Accept** to continue, otherwise click **Decline** to stop the installation process. You must accept the terms of the license agreement to continue the installation process.
7. If you already install the monitoring agent on this computer, a message might be displayed stating the installed version is newer than the agent installation, click **OK** to display the Select Features window.

8. Clear the **Tivoli Enterprise Monitoring Agents** check box.
9. Do the following thing, depending on which component you are installing application support.

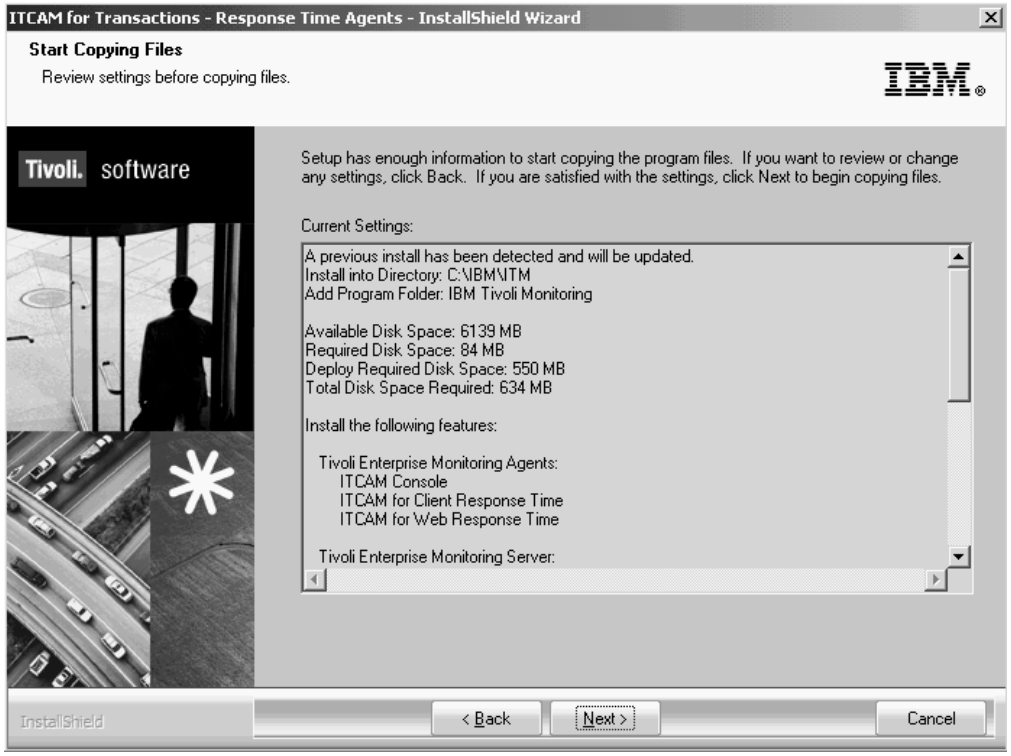
   Tip: You can choose to install support on one, two, or all the components installed on the same computer.

   - To install application support on Tivoli Enterprise Monitoring Server, select **Tivoli Enterprise Monitoring Server**.

     Important: If you have other components installed on the same computer, such as the Tivoli Enterprise Portal, also select those components to install the component-specific application support.
   - To install application support on Tivoli Enterprise Portal Server, select **Tivoli Enterprise Portal Server**.
   - To install application support on Tivoli Enterprise Portal Desktop Client, select **Tivoli Enterprise Portal Desktop Client** .
10. Click **Next** to display the Agent Deployment window. If you are installing locally, do not select any agents. If you are deploying an agent to a remote server, select the agent to enable remote deployment. See Chapter 11, “Working remotely,” on page 309 for further information.



11. Click **Next** to display the Response Time Information window.
12. Click **Next** to display the Start Copying Files window.

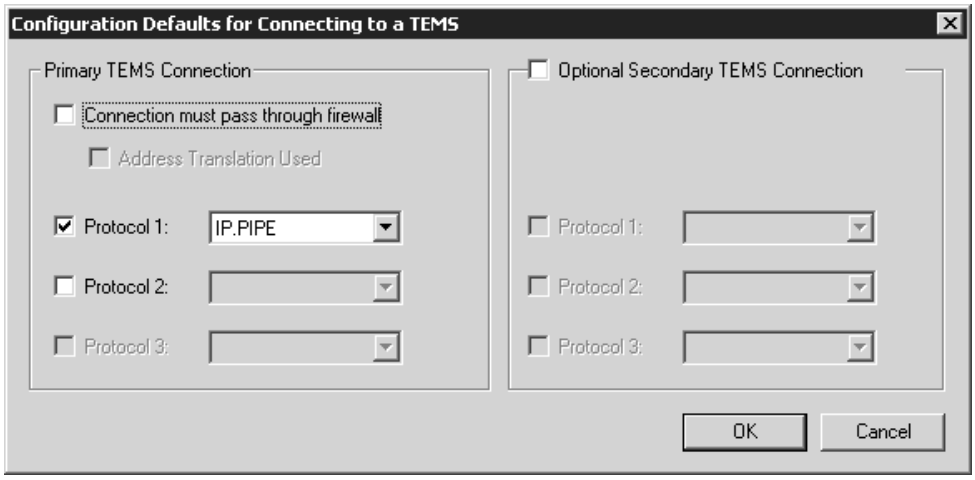


13. Click **Next** to continue, or click **Back** to modify your selection.

14. A message is displayed stating that you will not be able to cancel the installation or upgrade after this point. Click **Yes** to continue. Files will now be copied to your computer. This might take several minutes. A Setup Status window provides status messages about the installation progress.
15. Before the Setup is complete, a Setup Type window is displayed. Select the setup and configuration options to complete during installation. It is recommended to complete all setup and configuration during the installation, but you can clear the setup or configuration check boxes and complete the setup and configuration after the installation is complete. You cannot clear the check box that is preceded by an asterisk.

16. You might be prompted to enter the Tivoli Enterprise Portal Server host name. Type the host name of the system where the Tivoli Enterprise Portal Server is installed and click **Next**.

17. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:

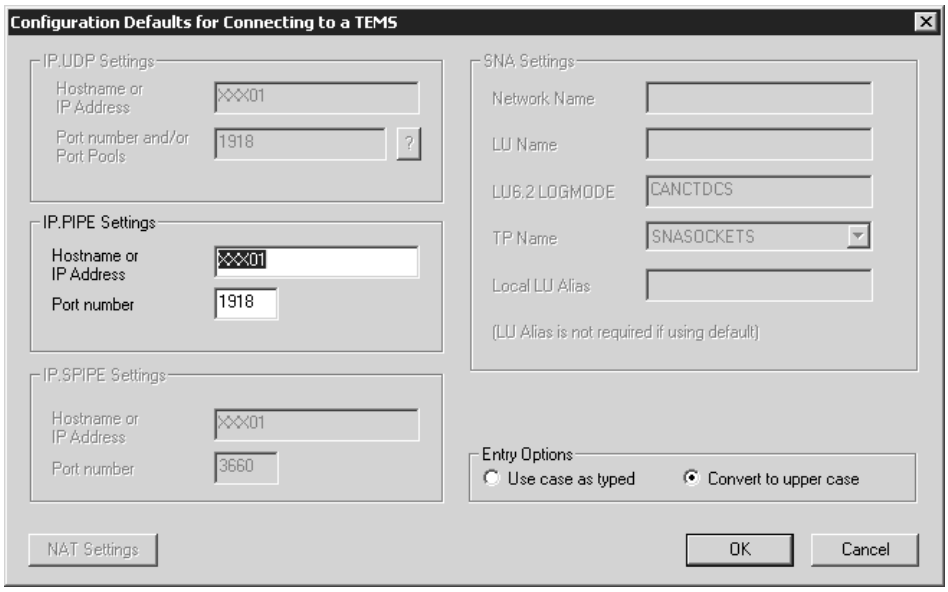


a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.
b. Identify the Information to collect before you begin installation and configuration that the agent uses to communicate with the monitoring server. You have four choices: `IP.UDP`, `IP.PIPE`, `IP.SPIPE`, or `SNA`.

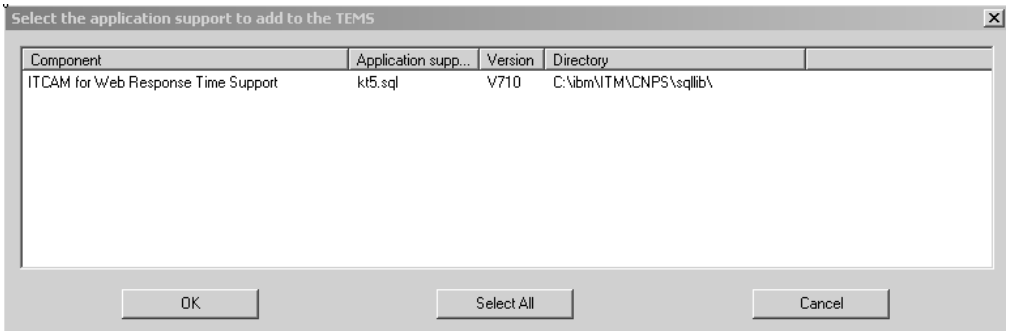
   Note: You can specify three communication methods. This enables you to set up backup communication methods. If the method you have identified as Protocol 1 fails, Protocol 2 is used.

c. Do not select **Optional Secondary TEMS Connection**. You can set up the standby support for agents after installation. See the IBM Tivoli Monitoring product documentation.

18. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**. The information on this window is automatically detected and depends on the protocol selected earlier.



19. You might be asked to specify the location of the monitoring server as either **on this computer** or **on a different computer**. Choose the appropriate option and click **OK**.
20. You might be asked to select the support you want to add to the Tivoli Enterprise Monitoring Server. Select all monitoring agents for which you want to install application support, and click **OK**.



21. The Application Support Complete window might be displayed containing details about the installation. Click **Next**.
22. *(Optional):* When installation is complete, the InstallShield Wizard Complete window opens. Clear the **Display the README File** check box if you do not want to view the readme file.
23. Click **Finish** to complete the installation.

Installing a monitoring agent on Linux or UNIX computers

Before you begin

Keep the following guidelines in mind:

- You are installing only **one** Application Management Console agent (which is also the robotic script file depot) in the IBM Tivoli Monitoring environment.
- You can install more than one Web Response Time agent on the same UNIX or Linux host, as long as the agents are installed in different file systems (for example, `/opt/IBM/ITM` and `/var/IBM/ITM2`). Some configuration customization is also required. For more information about installing agents, see "Installing multiple Web Response Time agents on the same Linux or UNIX host" on page 119.
- The Application Management Console must be on its own server-class box if monitoring more than 50 agents or if using Response Time to monitor a high volume web site generating thousands of transactions an hour. In test environments, it can be on the same computer as the Tivoli Enterprise Monitoring Server and the Tivoli Enterprise Portal.
- The Tivoli Enterprise Portal has a 26 character limitation for the host and application names in the Navigator. When you have a combination of names that are longer than 26 characters, the application name is truncated. For example, if you have application names such as *Websphere Plants* and *Websphere Petstore* on a computer with a long host name, the names might be displayed in the Navigator as `WebspherePl` and `WebspherePe`, whereas if the host name are only four characters shorter they are displayed as `WebspherePlants` and `WebspherePetsto`. To avoid truncating the host and application names in the Navigator, install the Application Management Console on a computer with a short host name.
- If you have a previously installed Response Time Tracking, version 6.1, you must uninstall it and verify that the following dll's were deleted before installing Robotic Response Time or Client Response Time.
  - `/usr/lib/libarm4.*`
  - `/usr/lib/libarmjni4.*`
  - `/usr/lib/libarmjni.*`
  - `/usr/lib/libarm32.*`
  - `/usr/lib/libarmcli.*`
- Install Rational Performance Tester software if you want to record and play back RPT test scripts on Robotic Response Time.
- If your environment has installed a `GSKit` version newer than 7.0.3.18, the installation fails. Uninstall the `GSKit` and try installing the agent again.
- If you are not logged in as root:
  - You cannot install or run Web Response Time. If you install the Web Response Time agent and attempt to start the kfcm120 process as a non-root user, the process will fail because it cannot open the NIC. Root is required to access the Network Interface.
  - You must have write access to the following:
    - *ITM_HOME* or the installation directory
    - `TEMP` directory for the system
    - The `JLOG` directory for the Tivoli Enterprise Monitoring agent, typically called `/var/ibm/tivoli/common/BWM/logs/`

  - Do not install two monitoring agents in the same *ITM_HOME* or installation directory using different user IDs. You can install two agents in the same *ITM_HOME* or installation direction if you use the same user ID for the installations.
- When installing Client Response Time (t4) or Robotic Response Time (t6), you must manually stop any applications that use ARM before starting the installation.

The following table provides an overview of what you need to for installation:

Table 20. Overview of installing a monitoring agent on Linux or UNIX computers

Steps	Where to find information
Install the monitoring agent.	"Procedure: Installing the agent "
Change the file permissions for files on the computer where you installed the agent, if you used a non-root user to install a monitoring agent	"Procedure: Changing file permissions" on page 118
Configure the monitoring agent.	"Configuring Client Response Time from the GUI (UNIX and Linux)" on page 152 "Configuring Application Management Console from the GUI (UNIX and Linux)" on page 146 "Configuring Robotic Response Time from the GUI (UNIX and Linux)" on page 160 "Configuring Web Response Time from the GUI (UNIX and Linux)" on page 170 Some agents require additional, agent-specific configuration parameters. See the agent documentation for the specific agents that you are configuring.
Start the monitoring agent.	"Starting and stopping monitoring agents" on page 326

Procedure: Installing the agent

1. In the directory where you extracted the installation files, run the following command: **./install.sh**.
2. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default (/opt/IBM/ITM) or type the full path to a different directory.
3. You are given a list of processes currently running. You are warned that these processes will be stopped and restarted during the installation. Type **1** and press **Enter** to continue the installation.

   A prompt similar to the following is displayed:

   ```
   Select one of the following:
   1) Install products to the local host.
   2) Install products to depot for remote deployment (requires TEMS).
   3) Exit install.

   Please enter a valid number:
   ```

4. Type **1** to start the installation and press **Enter**.

5. Type the number that corresponds to the language in which you want to display the software license agreement in and press **Enter**. (In some cases, you might not be prompted for this step. so you can skip it.)
6. Press **Enter** to display the agreement.
7. Type **1** to accept the agreement and press **Enter**.

   Note: Step 8 applies to agents installed from the IBM Tivoli Monitoring installation image, if the agent is installed from the agent installation, you can skip the step.
8. Type a 32 character encryption key and press **Enter**. This key was specified during the installation of the monitoring server to which this monitoring agent connects.
9. Type the number for the operating system on which you are installing and press **Enter**. The default value is your current operating system.
10. Type **1** to validate your selection and press **Enter**.
11. A list of available products for installation is provided. Type the number that corresponds to the monitoring agent or agents that you want to install and press **Enter**. To install more than one agent, use a comma (,) or a space to separate the numbers for each agent.
12. Type **1** to validate your selection and press **Enter** to start the installation.
13. When prompted to install additional products or product support packages, type **2** for **no** and press **Enter**.

Procedure: Changing file permissions

If you used a non-root user to install a monitoring agent on a UNIX computer, the file permissions are initially set to a low level. Run the following procedure to change these file permissions:

1. Log in to the computer as root, or become the root user by running the **su** command.
2. Create a new group (such as `itmusers`) to own all of the files in the IBM Tivoli Monitoring installation directory.

   For Linux, Solaris, and HPUX computers, run the following command: **groupadd itmusers**

   For an AIX computer, run the following command: **mkgroup itmusers**
3. Run the following command so that the `CANDLEHOME` environment variable correctly identifies IBM Tivoli Monitoring installation directory: **export CANDLEHOME = <ITM_Install_Directory>**

   CAUTION:
   Running the following steps in the wrong directory can change the permissions on every file in every file system on the computer.
4. Run the following command to change to the directory returned by the previous step: **cd $CANDLEHOME**
5. Run the following command to confirm that you are in the correct directory: **pwd**
6. Run the following commands:

   ```
   chgrp -R itmusers
   chmod -R o-rwx
   ```
7. Run the following command to change the ownership of additional agent files: **bin/SetPerm**

8. *(Optional)* To run the agent as a particular user, add the user to the `itmusers` group by editing the `/etc/group` file and adding the user to the `itmusers` group.

   For example, to run the agent as user `test1`, add following line is in the `/etc/group` file: `itmusers:x:504:test1`
9. Run the **su** command to switch to the user that you want to run the agent as or log in as that user.
10. Start the monitoring agent.

Installing multiple Web Response Time agents on the same Linux or UNIX host

On supported Linux and UNIX systems, multiple Web Response Time agents can run on the same system, as long as the agents are installed in different file systems (for example, `/opt/IBM/ITM` and `/var/IBM/ITM2`). Some configuration customization is also required.

Complete the following steps to achieve this setup:

1. Install multiple instances of the Web Response Time (WRT) agent in different file systems. These path names must not be subsets of each other. For example, for an installation deploying two WRT agents on the same box:
   - Correct:

     ```
     /opt/IBM/ITM
     /var/IBM/ITM2
     ```
   - Incorrect:

     ```
     /opt/candle
     /opt/candle/itcam_wrt/ServPoint
     ```
2. In the <*ITM_HOME*>/tmaitm6/wrm directory, add the following line to the `kfcmenv` file for each WRT agent, where <*port*> is an open port for the machine:

   ```
   KFC_API_MEDIASERVER_LISTEN_PORT=<port>;
   export KFC_API_MEDIASERVER_LISTEN_PORT
   ```

   Use port *12121* for the first agent, port *12122* for the second agent, port *12123* for the third agent, and so on.
3. In <*ITM_HOME*>/config, add this same line to the `t5.ini` and `t5.config` files for each WRT agent.
4. In <*ITM_HOME*>/config, add the following line to the `t5.config` and `t5.ini` files for each WRT agent. In <*ITM_HOME*>/tmaitm6/wrm, add the following line to the `kfcmenv` file for each WRT agent, where <*port*> is an open port on the system:

   ```
   KT5ERRORMESSAGEPORT=<port>
   ```

   Ensure that the port is unique for each agent.
5. In <*ITM_HOME*>/config, set the **CTIRA_HOST** variable in the `t5.config` and `t5.ini` files for each WRT agent to define the node name that the Tivoli Enterprise Portal (TEP) uses to identify each agent.

Avoid monitoring identical data: To avoid having each agent monitoring identical data, you must satisfy either of the following conditions:

- Each agent monitors a different NIC.
- In the <*ITM_HOME*>/tmaitm6/wrm/kfcmenv file for each WRT agent, use the following option to define which servers each WRT agent monitors:

  ```
  KFC_RESTRICT_HOST=<ip1>:<port1>,<ip2>:<port2>,...
  ```

Follow-up or related tasks

Complete the following tasks:

- Configure the agent you installed from either the command line or from the GUI:
  - "Configuring Response Time from the command line (UNIX and Linux) and agent configuration parameters" on page 177
  - "Configuring Client Response Time from the GUI (UNIX and Linux)" on page 152
  - "Configuring Robotic Response Time from the GUI (UNIX and Linux)" on page 160
  - "Configuring Web Response Time from the GUI (UNIX and Linux)" on page 170
- Chapter 12, "Configuring the Eclipse help server," on page 321

Install application support for Linux and UNIX

For each monitoring agent installed, you must install agent-specific application support on IBM Tivoli Monitoring components. You can choose to install application support on all components at one time, or you can install the application support on components separately. If installing components separately, the following order is recommended:

1. Tivoli Enterprise Monitoring Server
2. Tivoli Enterprise Portal server
3. Tivoli Enterprise Portal desktop client
4. Tivoli Enterprise Portal server browser client

The procedure for installing agent-specific application support on the components is nearly identical. The following steps provide the basic procedure and notes the differences in Table 21 on page 121. Repeat the procedure for each of the components on which you must install application support.

Note: The monitoring server is stopped during installation of application support on Tivoli Enterprise Monitoring Server. You should warn other users before taking action. If you do not stop the server, the software automatically stops it during the installation process.

Table 21. Application support installation summary by component type

For Tivoli Enterprise Monitoring Server application support	For Tivoli Enterprise Portal server	ForTivoli Enterprise Portal desktop client and browser support
1. Type the number that corresponds to `Tivoli Enterprise Monitoring Server support` 2. Type **1** to verify your selection and press **Enter**. A list of the components to install is displayed. 3. Type the number that corresponds to `all of the above` and press **Enter**. 4. Type **1** to verify your selection and start the installation and press **Enter** to begin the installation. 5. If the monitoring server is not started automatically, Start the Tivoli Enterprise Monitoring Server. 6. Run the following command to activate the application support on the monitoring server for the agent you installed: **./itmcmd support -t / <*tems_name*> <*pc*>** *tems_name* is the name of the Tivoli Enterprise Monitoring Server *pc* is the product code for the Response Time monitoring agent. See Table 11 on page 48 for a list of product codes. For example to install support for the Web Response Time agent on a monitoring server named `hub_itmdev17` run the following command: **./itmcmd support -t hub_itmdev17 t5** 7. Stop and then restart the Tivoli Enterprise Monitoring Server	1. Type the number that corresponds to `Tivoli Enterprise Portal server support` and press **Enter**. 2. Type **1** to verify your selection and press **Enter**. A list of the components to install is displayed. 3. Type the number that corresponds to `all of the above` and press **Enter**. 4. Type **1** to verify your selection and start the installation and press **Enter** to begin the installation. 5. Run the following command to configure the portal server with the new agent information: `./itmcmd config -A cq` 6. Complete the configuration as prompted. 7. Start the portal server.	1. Type the number that corresponds to `Tivoli Enterprise Portal Desktop Client support` or `Tivoli Enterprise Portal Browser Client support` 2. Press **Enter**. 3. Type **1** to verify your selection and press **Enter**. A list of the components to install is displayed. 4. Type the number that corresponds to `all of the above` and press **Enter**. 5. Type **1** to verify your selection and start the installation and press **Enter** to begin the installation. 6. Run the following command to configure the portal client with the new agent information: `./itmcmd config -A cj` Complete the configuration as prompted.

1. To allow the installation software to stop the component on which you are installing application support, go to step 22 on page 122. To manually stop the component, do one of the following:

- Stop the Tivoli Enterprise Monitoring Server

  The software asks for theIBM Tivoli Monitoring home directory.
- Stop the portal server
- Stop the Tivoli Enterprise Portal desktop client.

2. In the directory where you extracted the installation files, run the following command: **./install.sh**.
3. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default (/opt/IBM/ITM) or type the full path to the installation directory you used.

   The software displays the following prompt:

   ```
   Select one of the following:
   1) Install products to the local host.
   2) Install products to depot for remote deployment (requires TEMS).
   3) Exit install.

   Please enter a valid number:
   ```
4. Type **1** to start the installation and press **Enter**.
5. Type the number that corresponds to the language in which you want to display the software license agreement in and press **Enter**. (In some cases, you might not be prompted for this step. so you can skip it.)
6. Press **Enter** to display the agreement.
7. Type **1** to accept the agreement and press **Enter**.

   Note: Step 8applies to agents installed from the IBM Tivoli Monitoring installation image, if the agent is installed from the agent installation, you can skip the step.
8. Type a 32 character encryption key and press **Enter**. This key was specified during the installation of the monitoring server to which this monitoring agent connects.
9. Select the number corresponding to the component you are installing and press **Enter**.
10. Type **1** to verify your selection and press **Enter**.

    A list of the components to install is displayed.
11. Type the number that corresponds to all of the above and press **Enter**.
12. Type **1** to verify your selection and start the installation and press **Enter** to begin the installation.
13. If you want to install additional products or product support packages, enter **1** for **yes** or **2** for **no** and press **enter**.
14. Repeat this procedure for each of the components on which you must install application support.

Enabling File Transfer (UNIX and Linux)

Tivoli Enterprise Monitoring Server support limitation

Important: ***Do not*** install ITCAM File Transfer Enablement application support for an IBM Tivoli Composite Application Manager agent on Tivoli Enterprise Monitoring Server version 6.2.2 fix pack 2 or later. If you do, the monitoring server becomes inoperative, and you must repair it manually.

For ITCAM for Transactions version 7.3 and later, the version listed for File Transfer Enablement application support will always be shown as *07.21.00.00*.

Special considerations for installing on 64-bit and 32-bit versions for AIX

The ITCAM File Transfer Enablement (T1) has support for both 32-bit and 64-bit versions of the Tivoli Enterprise Monitoring Server on AIX. You must install the T1 support that matches your Tivoli Enterprise Monitoring Server version.

First determine if your Tivoli Enterprise Monitoring Server is running in 32-bit or 64-bit mode as follows:

1. Run Tivoli Monitoring's `cinfo` class on the Tivoli Enterprise Monitoring Server host:

   ITM_HOME`/bin/cinfo`
2. Choose the option to show installed products.
3. Find the entry similar to:

   ```
   ms Tivoli Enterprise Monitoring Server
       aix523  Version: 06.20.01.02
   ```
4. If the version ends in **3**, then you have a 32-bit version. For example, `aix523` and `aix533` are both 32-bit versions of the Tivoli Enterprise Monitoring Server.
5. If the version ends in **6**, then you have a 64-bit version. For example `aix526` and `aix536` are both 64-bit versions of the Tivoli Enterprise Monitoring Server.

Once you know the Tivoli Enterprise Monitoring Server version, you can choose the correct the File Transfer Enablement version (either 32-bit or 64-bit), when you run the ITCAM for Transactions install agent.

Enabling file transfer to ITCAM for Transaction agents

To enable file transfers to ITCAM for Transaction agents, you must install File Transfer Enablement (T1) support on the Tivoli Enterprise Monitoring Server and on each remote Tivoli Enterprise Monitoring Server. On Windows, this package is automatically installed as part of the agent installation, but for UNIX or Linux, you must explicitly select Tivoli File Transfer Enablement support as an option.

Follow these steps:

1. Access the installation software for the Application Management Console.
2. See “Install application support for Linux and UNIX” on page 120.
3. Select the operating system.
4. Select **ITCAM File Transfer Enablement** from the list of available packages.
5. Change the current directory to the T3 agent CD image.
6. Run the following command:

   ```
   # ./install.sh
   INSTALL
   ```
7. Enter the name of the IBM Tivoli Monitoring directory. The default is `/opt/IBM/ITM`

   The software displays the following message:

   ```
   ITM home directory "/opt/IBM/ITM" already exists.
   OK to use it [ y or n; "y" is default ]? y
   ```
8. Enter **Y**.

   The software displays the following message:

   ```
   The following processes are currently running:

   Product = Tivoli Enterprise Monitoring Server PID = 422128
   ```

```
Product = Tivoli Enterprise Portal Server PID = 573442
Product = Summarization and Pruning Agent PID = 368850
Product = Warehouse Proxy PID = 708836
install.sh warning: Existing products found to be running will be
  restarted during installation., continuing ...

Continue with this installation [ y or n; "y" is default ]?
```

9. Enter **Y**.

   The software displays the following message:

```
Continue with this installation [ y or n; "y" is default ]? y
Stopping TEMS...
TEMS stopped...
Stopping agent...
Agent stopped...
Stopping agent...
Agent stopped...
Stopping agent...
Agent stopped...
/opt/IBM/ITM


Select one of the following:

1) Install products to the local host.
2) Install products to depot for remote deployment (requires TEMS).
3) Exit install.

Please enter a valid number:
```

10. Enter **1**.
11. The software asks for the language you prefer. Enter the appropriate number.
12. Accept the software agreement.
13. Enter the number for the operating system and then confirm it.

    The software displays the following message:

```
The following products are available for installation:

1) ITCAM File Transfer Enablement V06.20.00.00
2) ITCAM for End User Response Time Dashboard Agent V06.20.00.00
3) Tivoli Enterprise Services User Interface V06.10.05.01
4) all of the above
Type the numbers for the products you want to install, or type "q" to
quit selection.
If you enter more than one number, separate the numbers by a comma or
a space.

Type your selections here:
```

14. Select **ITCAM File Transfer Enablement** and confirm your selection.
15. Restart the Tivoli Enterprise Monitoring Server and each remote Tivoli Enterprise Monitoring Server.
16. Perform this procedure on the Tivoli Enterprise Monitoring Server and each remote Tivoli Enterprise Monitoring Server.
17. Verify the installation:
    a. On each monitoring server, change the current directory to `ITM_HOME/bin`.
    b. Enter the command **`./cinfo`**.
    c. Select option **1** (Show products installed in this CandleHome).

       If File Transfer Enablement support is installed, you should see an entry like the following:

```
t1 ITCAM File Transfer Enablement
aix513 Version: 07.01.00.00
```

File Transfer Enablement is not a separate agent, so an error message displays if you attempt to configure, seed, or start/stop this entity. However, using the following commands does not harm the enterprise environment:

```
itmcmd config -A t1
itmcmd support -t <tems-name> t1
itmcmd agent { start | stop } t1
```

A status entry showing **stopped** for ITCAM File Transfer Enablement appears in the Manage Services GUI on UNIX and Linux platforms. Ignore this entry. Since File Transfer Enablement is not an agent, you cannot start this entity. Attempting to start File Transfer Enablement generates an error message, but causes no harm to your enterprise environment.

Verify installations of Response Time monitoring agents

These steps are recommended for verifying the installations of Response Time monitoring agents.

About this task

Complete the following steps to verify the installations of Response Time monitoring agents:

Procedure

1. Open the Manage Tivoli Enterprise Monitoring Services utility (if it does not open automatically) to see if the monitoring agent has been configured and started. The **Configured** column should say **Yes**.
2. *If* the value in the **Configured** column is blank and **Template** is in the **Task/Subsystem** column:
   a. Right-click **Template**
   b. Click **Configure Using Defaults**.
   c. Complete any windows requiring information by using the agent-specific configuration settings.

      Note: Do not enter non-ASCII characters on any of these windows. Entering characters from other character sets has unpredictable results.
3. Verify that the `kt*agent.exe` process is running on Windows and that the `kt*agent` is running on UNIX or Linux where `t*` is the product code.
   - On Windows, in the **Task Manager** check that the processes `kt*agent.exe` are running.
   - On UNIX, run: **`ps -ef | grep kt*`**
4. Verify the agent process is automatically started after system reboot.
   - Restart the virtual machine and perform Step 3.
5. Verify the t* agent is active in TEP where t* is the product code. See ITCAM for Transactions product codes for a list of product codes.
   a. Open the TEP and verify that you see the Application Management Console, Client Response Time, Robotic Response Time, and Web Response Time agents active in the navigator tree.

Installing Windows Network Monitor or Windows packet capture library

Before installing the Web Response Time agent on Windows platforms, you must install the Windows Network Monitor or WinPcap.

For all 64-bit Windows systems and Windows 2008, install WinPcap 4.1.1 or later. WinPcap can be found at http://www.winpcap.org.

For all other Windows systems, install the Windows Network Monitor by completing the following steps:

1. Do one of the following:
   - For Windows 2000, select **Start** > **Control Panel** > **Network and Dial-up Connections** > **Local Area Connection**.
   - For Windows 2003 and Windows XP, select **Start** > **Control Panel** > **Network Connections** > **Local Area Connection**.
2. Right click **Local Area Connection**.
3. In the pop-up menu, click **Properties**.
4. In the Local Area Connection Properties window, click **Install** if there is no Network Monitor Driver available.
5. Select **Protocol** from the Select Network Component window and click **Add**.
6. Select **Network Monitor Driver** from the Select Network Protocol window and click **OK**.
7. After the Network Monitor Driver is displayed on the Local Area Connections Properties window, click **Close**.

Performing a silent installation

This appendix provides information about installing Response Time agents using the silent installation method. This method of installation is useful for advanced users who prefer to input installation information once through a response file instead of repeatedly through an installation wizard.

You might run through the installation wizard one time to determine the values that you need to set for your monitoring needs and then use silent installation to install the rest of your environment. For more information about installing through the installation wizard, see "Installing Response Time monitoring agents and related software" on page 101.

The following table outlines the steps for performing a silent installation of Response Time agents.

Table 22. Silent Installation tasks

What to do	Where to find more information
Verify the software and hardware requirements for the agent you want to install have been met	Prerequisites
Collect the necessary information that you will asked for during installation and configuration	Information to collect before you begin installation and configuration
Run silent installation on Windows	"Perform a silent install of a monitoring agent (Windows)" on page 127

Table 22. Silent Installation tasks (continued)

What to do	Where to find more information
Run silent installation on UNIX or Linux	"Perform a silent install and configuration for a monitoring agent (UNIX or Linux)" on page 128
Install application support for the agent	"Install application support for Windows systems" on page 107 or "Install application support for Linux and UNIX" on page 120
Install support for the server	"Performing a server silent installation" on page 129
Start the monitoring agent	"Starting and stopping monitoring agents" on page 326

Perform a silent install of a monitoring agent (Windows)

The silent installation uses a response file, *install_directory*/Windows/silent.txt. For example, if you are using a CD-ROM, and the CD is D:\, then the path is D:\Windows\silent.txt . If you downloaded an installation zip file and unzipped it to C:\temp, then the path is C:\temp\Windows\silent.txt. The silent.txt specifies the necessary installation parameters.

Use the following steps to edit the silent.txt file as appropriate for your environment:

Attention: Do not modify any files that come with the installation (for example, the SETUP.ISS file) except the silent.txt file.

1. Locate the response file in the Windows subdirectory in the installation directory, and copy this file to a temporary directory on your system. For example, save the file as SILENT.TXT in the c:\temp directory.
2. Open your copy of the response file in a text editor.
3. Change the parameters as appropriate for your environment. Complete all of the steps listed in the file. Each line of the file must be either a comment (containing a semi-colon in column one) or a valid statement that starts in column one.

   Note: If you want to use the TCP/IP protocol, make sure to specify IP.UDP. If you specify TCP/IP, the installation uses IP.PIPE by default.
4. Save the file and close the editor.
5. Run the silent installation using one of the following methods:
   - "Running the silent installation from the command line with parameters"
   - "Using SMS" on page 128

Running the silent installation from the command line with parameters

Use the following steps to run the installation from the command line:

1. Start a DOS Command Shell.
2. From the shell, cd to the directory containing setup.exe and setup.ins (Usually is in WINDOWS subdirectory in the installation directory).
3. Run setup as follows. You must specify the parameters in the exact order as shown in the following example.

   ```
   start /wait setup /z"/sfC:\temp\SILENT.TXT" /s /f2"C:\temp\silent_setup.log"
   ```

Where

/z"/sf"
Specifies the name of the installation driver you customized for your site. This is a required parameter. This file must exist.

/s Specifies that this is a silent install. This causes nothing to be displayed during installation.

/f2 Specifies the name of the `InstallShield` log file. If you do not specify this parameter, the default is to create `Setup.log` in the same location as the `setup.iss` file (usually the `WINDOWS` subdirectory within the installation directory). In either case, the Setup program must be able to create and write to this file.

Using SMS

Use the following steps:

1. Copy the all the installation files to a LAN-based disk that SMS mounts on the desired computers. (Copy all files in the directory with `setup.exe` and `setup.ins` - usually the `WINDOWS` subdirectory within the installation directory.)
2. Replace the original `SILENT.TXT` file on the LAN disk with your modified version.
3. Edit the PDF file located with `setup.exe` and change the Setup invocation as follows:

   ```
   Setup /z"/sfC:\temp\SILENT.TXT" /s /f2"C:\temp\silent_setup.log"
   ```

Perform a silent install and configuration for a monitoring agent (UNIX or Linux)

The silent installation and configuration of a monitoring agent on Linux and UNIX requires an installation of code and a separate configuration. Both the installation and configuration use parameter files to define what you are installing and configuring.

The silent installation of the Response Time agent requires the information files `silent_install.txt` and `silent_config.txt`. The information file is provided in the installation directory. You must provide the necessary installation parameters as described in the silent installation and configuration files.

You can find more information about silent IBM Tivoli Monitoring installation in the *IBM Tivoli Monitoring: Installation and Setup Guide*.

1. To start a silent installation, run `install.sh`:

   ```
   ./install.sh -q -h <install_dir> -p <response_file>
   ```

   `<install_dir>` specifies the installation location for the monitoring agent. The default installation location is `/opt/IBM/ITM`.

   `<response_file>` identifies the response file that you edited to specify installation parameters, usually the `silent_install.txt` file. Specify the absolute path to this file.
2. To start a silent configuration, run the **`CandleConfig`** command in the `<install_dir>/bin` directory with the following configuration option:

   ```
   ./CandleConfig -A -p <response_file>
   ```

   `response_file` identifies the response file that you edited to specify configuration parameters, usually `silent_config.txt`. You can only find this

file after the agent is installed. Usually it is in the directory <install_dir>/config. Specify the absolute path to this file.

Performing a server silent installation

Provides instructions for the silent installation of a server.

Note: The following example is for Client Response Time (**T4**). Substitute the appropriate product names **T3** (Application Management Console), **T5** (Web Response Time), or **T6** (Robotic Response Time)

Windows

Location of the silent response file: WINDOWS\silent.txt on the product CD or download image for each agent. It has the following lines that control what gets installed:

```
KGLWICMA=Tivoli Enterprise Monitoring Agent Framework
;
;KT4WICMS=ITCAM for Client Response Time Support ( TEMS )
;KT4WIXEW=ITCAM for Client Response Time Support ( TEP Workstation )
;KT4WICNS=ITCAM for Client Response Time Support ( TEP Server )
;KT4WICMA=ITCAM for Client Response Time ( TEMA )
```

To install Tivoli Enterprise Monitoring Server server support, uncomment the line, ;KT4WICMS=ITCAM for Client Response Time Support (TEMS).

To perform the installation, run the silent installer the same way that you would for an monitoring agent installation. See "Perform a silent install of a monitoring agent (Windows)" on page 127.

UNIX **Location of the silent response file**: /silent_install.txt on the product CD or download image for each agent. To install Tivoli Enterprise Monitoring Server server support, make sure the following lines are in the file:

```
INSTALL_FOR_PLATFORM=tms
INSTALL_PRODUCT=all
```

To perform the installation, run the silent installer the same way that you would for an monitoring agent installation. See "Perform a silent install and configuration for a monitoring agent (UNIX or Linux)" on page 128.

Installing Rational products

You install IBM Rational products to work with the Robotic Response Time monitoring agent.

This section describes the following tasks:

- "Installing integration support for Rational Performance Tester" on page 130
  You must already have installed Rational Performance Tester version 8.0 or later.
- "Installing integration support for Rational Functional Tester" on page 134
  You must already have installed Rational Functional Tester version 8.1 or later.
- "Installing Rational Robot" on page 135
- "Removing integration support" on page 138

Considerations for Rational Performance Tester v8.2

This section includes some additional information you might need when using IBM Rational Performance Tester version 8.2.

If you are upgrading Rational Performance Tester to version 8.2 to work with ITCAM for Transactions version 7.3, follow this general procedure:

1. Export your existing Rational Performance Tester scripts so they can be later imported into Rational Performance Tester version 8.2. When you do this task, be sure to select the option, **Test Assets with Dependencies**. See the following link for more information:

   `http://publib.boulder.ibm.com/infocenter/rpthelp/v8r2m0/index.jsp?topic=/com.ibm.rational.test.lt.doc/topics/tcopydeps.html`
2. Uninstall the existing version of Rational Performance Tester before installing version 8.2. You can use the Windows Add/Remove Programs control panel to uninstall Rational Performance Tester. Clean up the Windows registry and the Windows file system manually as described in the following technote:

   `http://www-01.ibm.com/support/docview.wss?uid=swg21283462`
3. Refer to the following technote for additional information about installing Rational Performance Tester 8.2 so that you are automatically granted a permanent license. This technote also provides information about the ITCAM export plug-in installations as part of Rational Performance Tester. You do not need to install a license manager.

   `http://www-01.ibm.com/support/docview.wss?uid=swg21502172`
4. After you uninstall the previous version of Rational Performance Tester and install version 8.2, remember to reboot the current workstation or server.
5. After you import your scripts into Rational Performance Tester version 8.2, run a test on each script.
6. Export the scripts from Rational Performance Tester version 8.2 to the ITCAM for Transactions Application Management Console.

Installing integration support for Rational Performance Tester

This section describes the procedure for installing integration support for IBM Rational Performance Tester for use with the Robotic Response Time monitoring agent.

Use Rational Performance Tester to record and upload robotic script tests when monitoring HTTP or HTTPS web applications, SAP, Siebel, or Citrix applications for performance.

Rational Performance Tester can create tests of web pages without manual coding of the verification points that are required in Rational Robot VU. No programming knowledge is necessary to create, comprehend, modify, and execute a performance test. A test created with Rational Performance Tester provides a graphical illustration of the web pages visited during execution. Code editing is unnecessary to create a multiuser test. For more advanced testers, information about items such as underlying page elements and server responses is also available.

Creates, executes and analyzes tests to validate the reliability of complex e-business applications.

Note: You cannot migrate previously recorded Rational Robot VU scripts to Rational Performance Tester scripts. You must record new scripts using Rational Performance Tester.

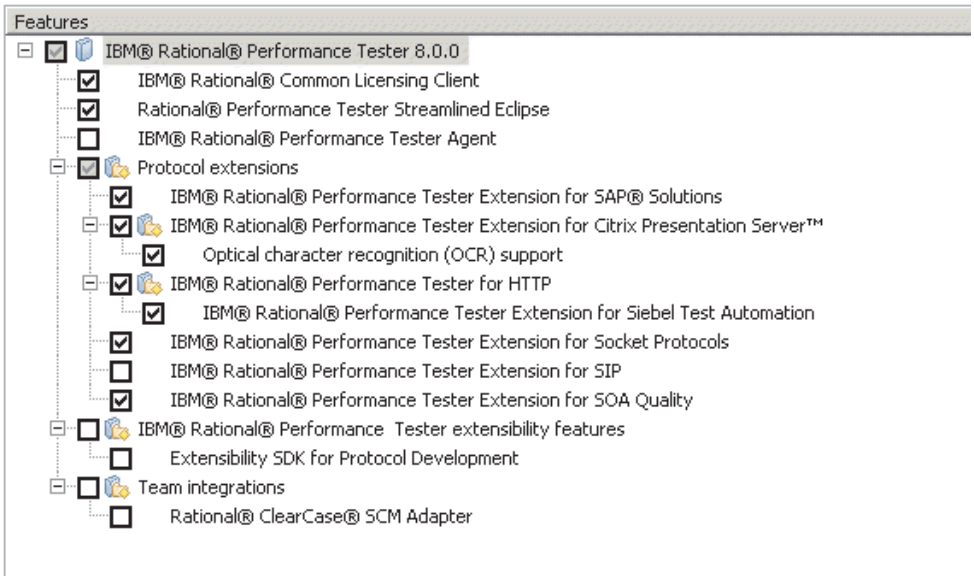
Installation restrictions

Observe the following restrictions when installing integration support for Rational Performance Tester:

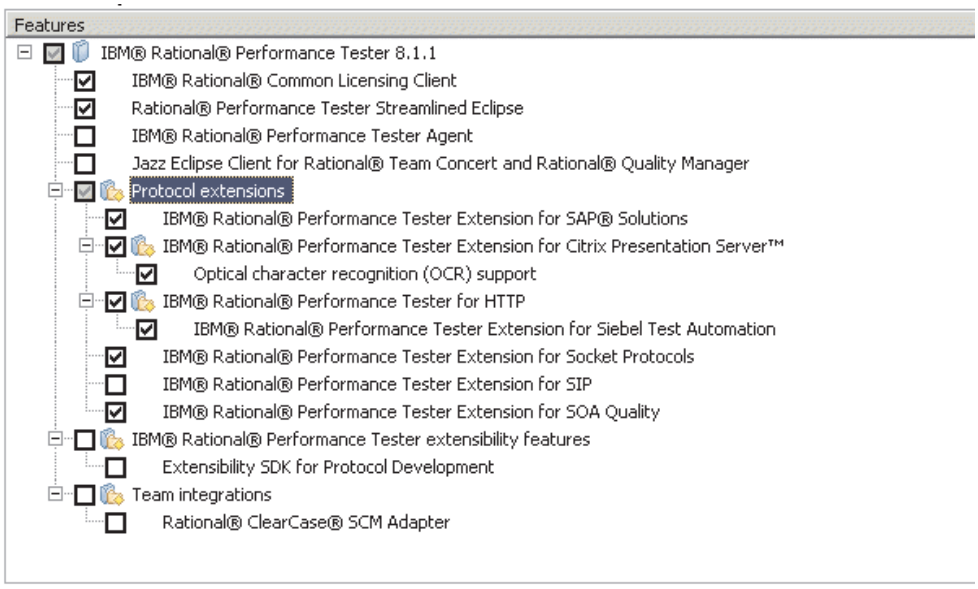
- You must already have Rational Performance Tester version 8.0 or later installed on your system.
- You can install Rational Performance Tester on any Windows system from which you want to record the tests *except* on the system where the Robotic Response Time agent is installed.
- If for some reason you change the directory path for the Rational Performance Tester wrapper installer and you rename the *disk1* folder, do not include the word **disk** in the directory name, or the installation fails.
- The system on which you install Rational Performance Tester must be able to connect directly to the system where the Application Management Console is installed. Ensure that there is no firewall between the Rational Performance Tester system and the Application Management Console system.
- If you install Rational Performance Tester and Robotic Response Time or Client Response Time on the same Windows system, you must remove the ITCAM RTT 6.1 Management Agent, which Rational Performance Tester installs automatically. Use the Windows Control Panel **Add or Remove Programs** function to remove this agent.

  After you remove this agent, Rational Performance Tester can coexist with the Robotic Response Time and Client Response Time monitoring agents, however you lose some functions, such as collecting performance data from the workbench.
- The ITCAM for Transactions Rational Integration Support Export Plugin depends on Rational Performance Tester protocol extensions being installed. The following protocol extensions must be selected during Rational Performance Tester installation. If Rational Performance Tester was installed without selecting these protocol extensions, you can modify the installation using the Modify option of the Installation Manager program, and selecting the protocol extensions that are missing.

  Select the following protocol extensions for Rational Performance Tester version 8.0:



Select the following protocol extensions for Rational Performance Tester version 8.1:



Except for the SIP protocol extension, all other protocol extensions are required for use with the ITCAM for Transactions Export Plugin. Note that if Rational Performance Tester is being installed on a computer where you also plan to install the Robotic Response Time agent, do not select the *IBM Rational Performance Tester Agent* option, because it causes conflicts in the data collection modules.

Installing integration support

Run the Rational Integration Support Installer to upgrade your existing version of Rational Performance Tester to support the Robotic Response Time agent, and install necessary interim fixes.

Note: Be sure to remove previous interim fixes from your dropins folder before upgrading your Rational Performance Tester installation.

To install integration support for Rational Performance Tester, complete the following procedure:

1. Verify the supported platforms for Rational Performance Tester at `http://www-306.ibm.com/software/awdtools/tester/performance/sysreq/index.html`.
2. **Important:** Remove previous interim fixes from your dropins folder before installing this integration support.
3. Access the Rational Integration Support installation program from the installation media (CD or downloaded image).
4. Navigate to the disk1 folder and run **`setup.exe`** to start the Rational Integration Support Installer program.
5. View the Welcome information and click **Next**.
6. Accept the terms of the license agreement and click **Next**.
7. If you do not have Rational Performance Tester version 8.0 or later already installed, you will receive an error message and will not be able to continue. Otherwise, select the **IBM Rational Performance Tester Version 8.1.0** check box and click **Next**.

8. Confirm your selection to install the ITCAM for Transactions Version 7.2 Integration Plug-in for Rational Performance Tester Version 8.1.0 (along with the License Key and interim fixes) and click **Next**.
9. The installation of the license key, integration support, and hot fixes is started. Wait for completion.
10. When the installation completes, a message indicating success or failure is displayed. Click **Finish** to close the Rational Integration Support Installer program.
11. Examine the following log file for more information: `C:\Program Files\IBM\tivoli\common\BWM\logs\trace-install.log` for the result. If the installation is successful, the end of the log contains the message: `The installation program finished successfully.` If the installation fails, the end of the log contains some error information about why the installation failed.

Silently installing integration support for Rational Performance Tester

To install integration support for Rational Performance Tester silently, complete the following steps:

1. Verify that the file `$WrapperImagerDir\disk1\runtime\RationalResponse.properties` contains the following line:

   ```
   InstallRPT = true
   ```

   If the line is commented out with the # character in column 1, remove the # character.
2. Change to the directory: `$WrapperImagerDir\disk1`
3. Run the command:

   ```
   start /wait setup.exe -silent -options runtime\RationalResponse.opt
   ```
4. After the silent installation completes, check the file: `C:\Program Files\IBM\tivoli\common\BWM\logs\trace-install.log` for the result. If the installation is successful, the end of the log contains the message: `The installation program finished successfully.` If the installation fails, the end of the log contains some error information about why the installation failed.

Finding more information

See the following links for more information about IBM Rational Performance Tester:

- Features:

  ```
  http://www-306.ibm.com/software/awdtools/tester/performance/
  ```
- Product documentation:

  ```
  version 8.1: http://publib.boulder.ibm.com/infocenter/rpthelp/v8r1m0/index.jsp
  version 8.0: http://publib.boulder.ibm.com/infocenter/rpthelp/v8r0m0/index.jsp
  ```

Installing integration support for Rational Functional Tester

This section describes the procedure for installing integration support for Rational Functional Tester for use with the Robotic Response Time monitoring agent.

IBM Rational Functional Tester is an automated tool for functional testing, regression testing, GUI testing, and data-driven testing. The following list describes some of its key features:

- Automated tests can play back reliably even if the user interface for your application changes frequently.
- An automated wizard for data-driven testing increases test coverage by reusing individual tests with multiple sets of test data.

  Tip: If you use Rational Functional Tester as a test tool during the development cycle, you can reuse the scripts for production monitoring with Robotic Response Time.
- Storyboard testing simplifies test visualization and editing using natural language and rendered screen shots.
- Dynamic data is validated with multiple wizards, verification points, and support for regular expression patterns.
- Keywords can be used to automate portions of manual tests.
- Supports and extends the functionality of Rational Robot GUI.

Installing integration support

Run the Rational Integration Support Installer to upgrade your existing version of Rational Functional Tester to support the Robotic Response Time agent, and install necessary hotfixes.

To install integration support for Rational Functional Tester, complete the following procedure:

1. Verify the supported platforms for Rational Functional Tester at `http://www-306.ibm.com/software/awdtools/tester/functional/sysreq/index.html`.

   Note: Even though Rational Functional Tester is supported on Windows 2008 and Windows Vista, installation of this Rational Integration Support for Rational Functional Tester is not supported on those operating systems for this release.
2. Access the Rational Integration Support Installer program from the installation media (CD or downloaded image).
3. Navigate to the disk1 folder and run **`setup.exe`** to start the Rational Integration Support Installer program.
4. View the Welcome information and click **Next**.
5. Accept the terms of the license agreement and click **Next**.
6. If you do not have Rational Functional Tester version 8.1 or later already installed, you will receive an error message and will not be able to continue. Otherwise, select the **IBM Rational Functional Tester Version 8.1.0** check box and click **Next**.
7. Confirm your selection to install the ITCAM for Transactions Version 7.2 Integration Plug-in for Rational Functional Tester Version 8.1.0 and click **Next**.
8. The installation of integration support starts. Wait for completion.

9. When the installation completes, a success message is displayed. Click **Finish** to close the Rational Integration Support Installer program.

Silently installing integration support for Rational Functional Tester

To install integration support for Rational Functional Tester silently, complete the following steps:

1. Verify that the file `$WrapperImagerDir\disk1\runtime\RationalResponse.properties` contains the following line:

   ```
   InstallRFT = true
   ```

   If the line is commented out with the # character in column 1, remove the # character.
2. Change to the directory: `$WrapperImagerDir\disk1`
3. Run the command:

   ```
   start /wait setup.exe -silent -options runtime\RationalResponse.opt
   ```
4. After the silent installation completes, check the file: `C:\Program Files\IBM\tivoli\common\BWM\logs\trace-install.log` for the result. If the installation is successful, the end of the log contains the message: `The installation program finished successfully.` If the installation fails, the end of the log contains some error information about why the installation failed.

Follow-up or related tasks

See the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide* for additional information on using Rational Functional Tester with Response Time.

Finding more information

See the following links for more information about IBM Rational Functional Tester:

- Features: http://www.ibm.com/software/awdtools/tester/functional/index.html
- Product documentation:

  ```
  Rational Functional Tester Information Center
  ```

Installing Rational Robot

Robotic Response Time uses Rational Robot to record and playback the user experience on Windows based applications and web based applications. Install Rational Robot if you are monitoring Windows GUI applications or if you are using Rational Robot VU HTTP legacy support. You can install Rational Robot on any computer with the following characteristics:

- The computer runs one of the supported Windows versions.
- The computer can connect to the web resources needed for the robotic transaction.
- The computer is accessible to the management server for the uploading of completed recordings.

For more information about working with the Generic Windows GUI and Generic Windows VU components, refer to the online help. For more information about prerequisites for Rational Robot, see Prerequisites.

Before you begin

Note: Rational Robot is no longer shipped and available as part of ITCAM for Transactions version 7.3 and later download packages, but you can still upload Rational Robot scripts using MFU, and use Rational Robot scripts for playback on Robotic Response Time agents if you have a currently supported version of Rational Robot.

If you plan to run Rational Robot on a Citrix server, contact IBM Software Support and request a floating Rational Robot License key before starting the installation.

You must have access to an bulk-file transfer program, such as FTP, and a file extraction program, such as `WinZip` or `PKZIP`.

Procedure

1. Insert the CD containing Rational Robot.
2. Run the `setup.exe` command to start the installation wizard.
3. On the **Welcome to the Setup Wizard** window, click **Next** to display Product Selection window.
4. Select **Rational Robot** and click **Next**.
5. Select your deployment method and click **Next**.
6. Click **Next**.
7. Close any open applications and then click **Next**.
8. Click **read non-IBM terms** to read the license agreement.
9. Click on **Accept**, and then click **Next**.
10. Click **Next** to accept the location.
11. Select the programs you want to install and click **Next**.
12. Click **Install**.
13. Select **Import a Rational License File** and click **Next**.
14. Select the `ibm_robot.upd` file.
15. Click **Import**.
16. Click **Import** again.
17. Click **OK**.
18. Close the window.
19. Restart the computer.

Follow-up or related tasks

Set DCOM Config Security permissions for the Administrator. The distributed component object model (DCOM) is a network extension of the component object model (COM) technology that enables interprocess communication across the network on Windows systems. DCOM enables communication across the network on Windows. The default access and launch permissions of DCOM do not give the Rational Robot Player account permission to launch Microsoft Internet Explorer. You can add the Rational Robot Player account to the launch and access permissions for DCOM by using the `DCOMCNFG.EXE` utility.

1. Restart the managed system after installing Rational Robot.
2. Set DCOM Config Default Security permissions for the user on the management agent on which Rational Robot is installed:

a. Using Windows Explorer, navigate to the directory where the ***DCOMCNFG.EXE*** file is located. Usually this is in *system_root*\system32, where *system_root* is the drive and directory where the Windows system is installed (for example, C:\Winnt).

b. Run the ***DCOMCNFG.EXE*** file.

c. Do one of the following:

For Windows 2000 and XP	For Windows 2003
1. Click the **Default Security** tab. 2. Click **Edit Default** in the **Default Access Permissions** group.	1. From the navigation tree click **Console Root -> Component Services -> Computers -> My Computer -> DCOM Config -> Internet Explorer**. 2. Right click **Properties** and select the **Security** tab 3. Click **Customize -> Edit** in the **Access Permissions groups**.

d. Click **Add**.

e. Select the computer name from the **List Names From** list.

f. Click **Show Users**.

g. Do one of the following:

Select an existing user	Create a new user
1. Select the Agent robotic user from the list of names.	If the management agent user is not in the list: 1. Click **Add** to display the Add Users and Groups window. 2. Select the user from the list. 3. Click **Add** and click **OK**.

h. Choose **Allow Access** from the Type of Access drop-down menu and click **OK**.

i. Do one of the following:

For Windows 2000 and XP	For Windows 2003
1. Click **Edit Default** in the **Default Access Permissions** group.	1. Right click **Properties** and select the **Security** tab 2. Click **Customize -> Edit** in the **Access Permissions groups**.

j. Select the user from the list and choose **Allow Launch** from the Type of Access drop-down menu.

k. Click **OK**.

l. Do one of the following:

For Windows 2000 and XP	For Windows 2003
1. Click **Edit Default** in the **Default Access Permissions** group.	1. Right click **Properties** and select the **Security** tab 2. Click **Customize -> Edit** in the **Access Permissions groups**.

m. Select the user from the list.

n. Choose **Full Control** from the **Type of Access** drop-down menu
o. Click **OK** twice to finish the procedure.

Removing integration support

Follow these steps to uninstall integration support for Rational Performance Tester or Rational Functional Tester:

1. Before uninstalling integration support, close all editors and switch out of the ITCAM for Transactions perspective in the Eclipse workbench.
2. Select **Add or Remove Programs** from the Windows Control Panel.
3. Select the entry for the **ITCAM for Transactions 7.2 Integration Plug-in**.
4. Click **Change/Remove**.
5. You can see the results of this task in the following log file: `%temp%\uninstall_rt_plugins.log` (for example, `C:\Documents and Settings\Administrator\Local Settings\Temp\uninstall_rt_plugins.log`).

If you still see the ITCAM for Transactions perspective in the workbench, right-click the perspective and close it manually, or restart the workbench using the **`-clean`** option.

Uninstalling a monitoring agent

Use the procedures in this section to uninstall a Response Time monitoring agent from your computer.

Note: When you uninstall a monitoring agent, the uninstallation process might leave behind directories or files that have been added or modified by users. You must manually delete the following files: `\IBM\Rational` and `..\IBM\SDP70Shared`.

Manually removing HTTPS configuration before uninstalling the Web Response Time agent

Before uninstalling the Web Response Time monitoring agent, complete the following manual steps to remove HTTPS configuration from the httpd.conf file:

1. Stop IHS.
2. Navigate to the `C:\IBM\HTTPServer\conf` directory, and locate the httpd.conf file.
3. Edit this file using your preferred text editor, and manually remove the following lines from the file:

```
LoadFile C:\IBM\ITM\tmaitm6\wrm\analyzer\kbb.dll
LoadFile C:\IBM\ITM\tmaitm6\wrm\analyzer\klx.dll
LoadModule candle_kfci_module C:\IBM\ITM\tmaitm6\wrm\analyzer\kfciihs7.dll
```

4. Save the httpd.conf file.
5. Proceed with the manual uninstallation of the Web Response Monitor (see the next section) and then uninstall the Web Response Time monitoring agent.

Manually uninstalling the Web Response Monitor before uninstalling the Web Response Time agent

Before uninstalling the Web Response Time monitoring agent on Windows systems, complete the following manual steps to uninstall the Web Response Monitor:

1. Stop the Web Response Time monitoring agent.

2. From a command prompt, navigate to the `wrm_images\install` directory, typically located at

   ```
   %ITM_Home%\tmaitm6\wrm_images\install
   ```

3. Run the following command:

   ```
   setup.exe /s /f1%ITM_HOME%\tmaitm6\wrm_images\install\remove.iss
   /zuninstall /verbose\uninstall_wrm.log
   ```

4. After this command completes, proceed with the uninstallation of the Web Response Time monitoring agent.

Manually uninstalling the Client Application Tracker before uninstalling the Client Response Time agent

Before uninstalling the Client Response Time monitoring agent on Windows systems, complete the following manual steps to uninstall the Client Application Tracker:

1. Stop the Client Response Time monitoring agent.
2. From a command prompt, navigate to the `cat_image` directory, typically located at

   ```
   %ITM_Home%\tmaitm6\cat_image
   ```

3. Run the following command:

   ```
   setup.exe
   ```

4. After this command completes, proceed with the uninstallation of the Client Response Time monitoring agent.

Use the following procedures to uninstall the monitoring agent:

- For Windows operating systems:
  1. Stop all Internet Service Monitoring processes.
  2. From the desktop, click **Start** > **Settings** > **Control Panel** (for Windows 2000) or **Start** > **Control Panel** (for Windows 2003).
  3. Click **Add or Remove Programs**.
  4. Do one of the following things depending on what type of image you used to install the specific agent most recently.
     - If the agent is installed with the agent-specific image, click the agent name, such as IBM Tivoli Composite Application Manager Console and IBM Tivoli Composite Application Manager for Robotic Response Time.
     - If the agent is installed with the integrated image, click **ITCAM for Transactions - Response Time Agents**.

       Remember: One integrated image to install all Response Time agents is provided by ITCAM for Transactions version 7.2, instead of separate images for each agent in previous versions. For example, if you installed a previous version of Robotic Response Time and upgrade it to version 7.2 with the integrated image, you should select **ITCAM for Transactions: Response Time Agents** in this step.
     - If the agent is installed or upgraded in IBM Tivoli Monitoring version 6.2.1 or later, you might not find agent-specific entries listed under **Add or Remove Programs**. In this case, select **IBM Tivoli Monitoring** and follow the on-screen prompts to select the specific agent or agents to uninstall.
  5. Click **Change/Remove**.
  6. Select **Remove** and click **Next**.

Important: If you select **ITCAM for Transactions - Response Time Agents** in the previous step, selecting **Remove** in this step will uninstall **all** the Response Time agents that are installed with the integrated image; if you only want to uninstall some specific agent, select **Modify** in this step and clear the check box next to the agent on the following Add or Remove Features window.

7. Click **OK** to confirm the uninstallation.
8. Click **Finish**.

To perform a silent uninstall, modify the `silent.txt` file by commenting all the lines, except for the following:

```
[ACTION TYPE]
REMOVEALL=Yes
```

For more information on `silent.txt`, see "Performing a silent installation" on page 126.

- For UNIX or Linux operating systems:
  1. Stop all Internet Service Monitoring processes.
  2. From a command prompt, run the following command to change to the appropriate `/bin` directory:

     `cd` *`install_dir`*`/bin`

     where *`install_dir`* is the path for the home directory for the agent.
  3. Run the following command:

     `./uninstall.sh`

     A numbered list of product codes, architecture codes, version and release numbers, and product titles is displayed for all installed products.
  4. Type the number for the monitoring agent you want to uninstall. Repeat this step for each additional installed product you want to uninstall.

The following instructions tell how to verify the removal (uninstallation) of a monitoring agent from Windows, UNIX, or Linux systems.

Verify monitoring agent removal from Windows	Verify monitoring agent removal from UNIX and Linux
1. Verify the service, including processes, are removed. • Go to **Control Panel - Administrative Tools - Services** and be sure the agent does not exist in the Services list. • Go to the **Windows Task Manager** and verify the agent is not running. For Web Response Time, you need to verify that the Analyzer (kfcm120) process is not running. 2. Verify the related files are removed. 3. Verify the agent node on the Tivoli Enterprise Portal is removed. • If the agent node is still installed on the Tivoli Enterprise Portal, you will see a Navigator window on the left side of the Tivoli Enterprise Portal window. • If the agent node is offline, you must manually remove it by right-clicking the node in the Navigator window and selecting **Remove**.	1. Verify the service, including processes, are removed. • From the command line, run `CANDLE_HOME/bin/cinfo -r` or run `ps -if | grep kt*agent` where *t** is the *product code*. 2. Verify the related files are removed. 3. Verify the agent node on the Tivoli Enterprise Portal is removed. • If the agent node is still installed on the Tivoli Enterprise Portal, you will see a Navigator window on the left side of the Tivoli Enterprise Portal window. • If the agent node is offline, you must manually remove it by right-clicking the node in the Navigator window and selecting **Remove**.

Chapter 7. Configuring Response Time and related software

Administrators can configure Response Time and related software to meet the specific requirements of their environment.

Configuring Application Management Console

This section includes the following topics:

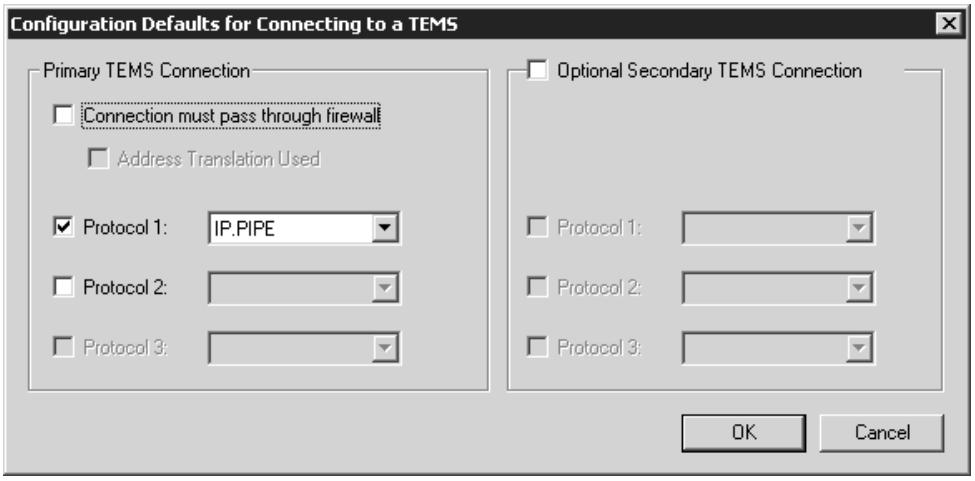
- "Configuring Application Management Console (Windows)"
- "Configuring Application Management Console from the GUI (UNIX and Linux)" on page 146

Note: In addition to these methods, you can also configure monitoring agents from the command line. See "Configuring Response Time from the command line (UNIX and Linux) and agent configuration parameters" on page 177

Configuring Application Management Console (Windows)

Administrators must configure Application Management Console to meet the specific requirements of their environment.

1. Make sure that you understand and have collected the configuration parameters for this agent from "Application Management Console parameters" on page 178 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. If you selected **Next** from step 16 (in the section called "Install a monitoring agent on Windows systems" on page 102) the installation software displays the Tivoli Enterprise Monitoring Server Configuration window.
3. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:



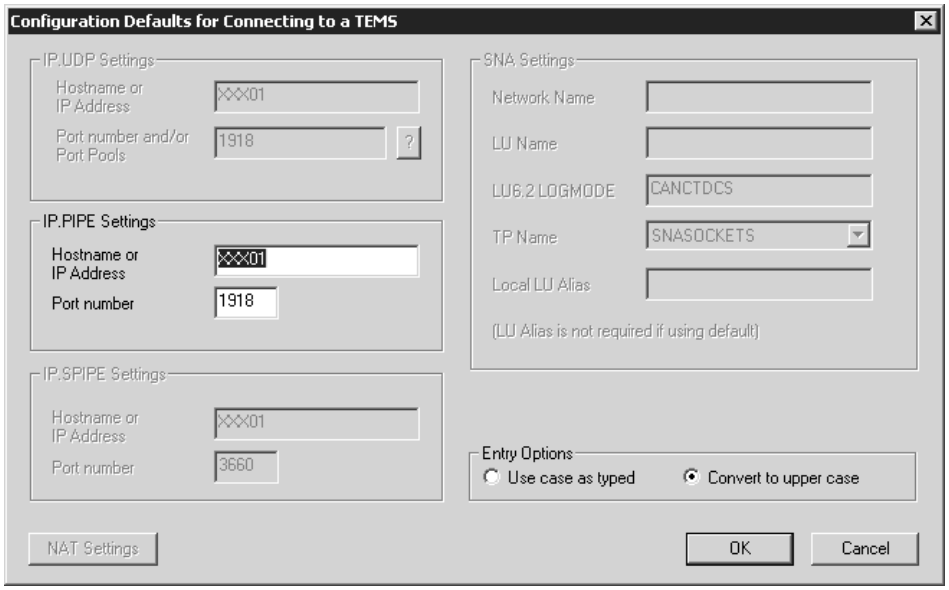
   a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.

b. Identify the Information to collect before you begin installation and configuration that the agent uses to communicate with the monitoring server. You have four choices: `IP.UDP`, `IP.PIPE`, `IP.SPIPE`, or `SNA`.

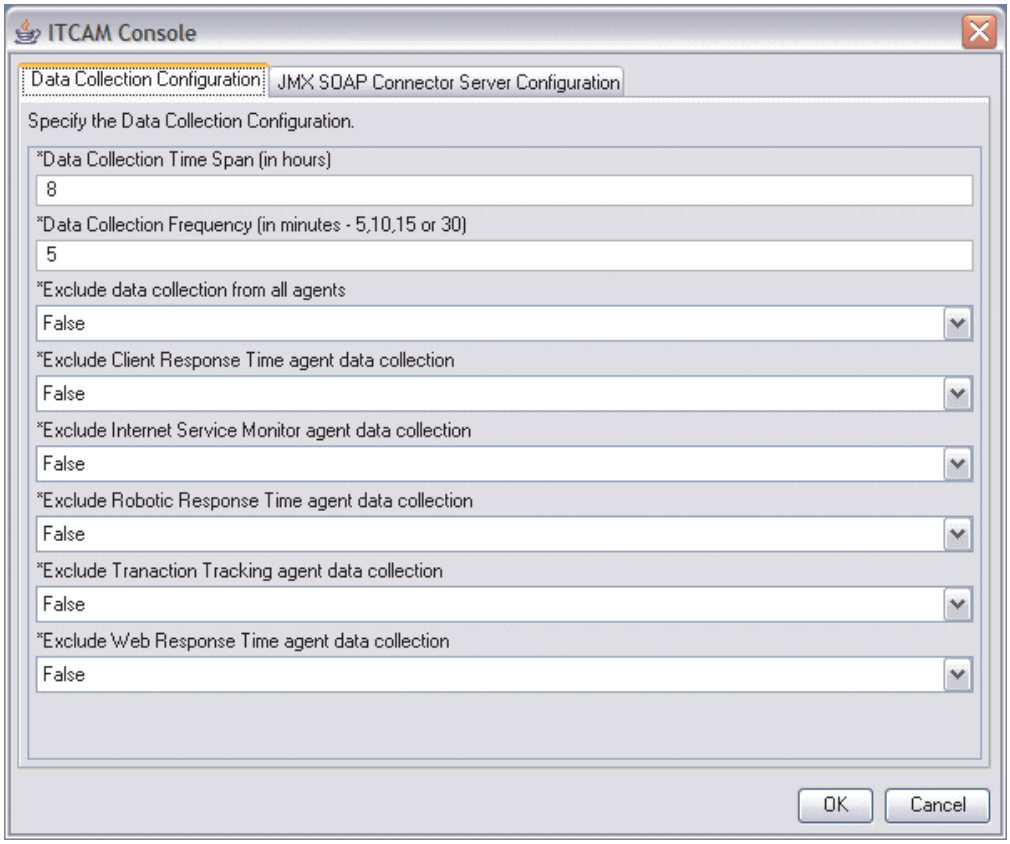
   Note: You can specify three communication methods. This enables you to set up backup communication methods. If the method you have identified as Protocol 1 fails, Protocol 2 is used.

c. Do not select **Optional Secondary TEMS Connection**. You can set up the standby support for agents after installation. See the IBM Tivoli Monitoring product documentation.

4. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**. The information on this window is automatically detected and depends on the protocol selected earlier.



5. In the Configuration window, select the **Data Collection Configuration** tab.



6. Specify the configuration information about how data is analyzed.

   The Application Management Console agent collects its data from other ITCAM for Transactions agents on a regular basis. It does this data collection automatically with no configuration required by using the KT1 File Transfer Enablement service installed on the monitoring service.

   By default, the data is collected from the agents every 5 minutes except for the Transaction Reporter, from which the data is collected every 2 minutes to ensure not missing any data. After the data is collected, the metrics in the Application Management Console are updated with the latest values.

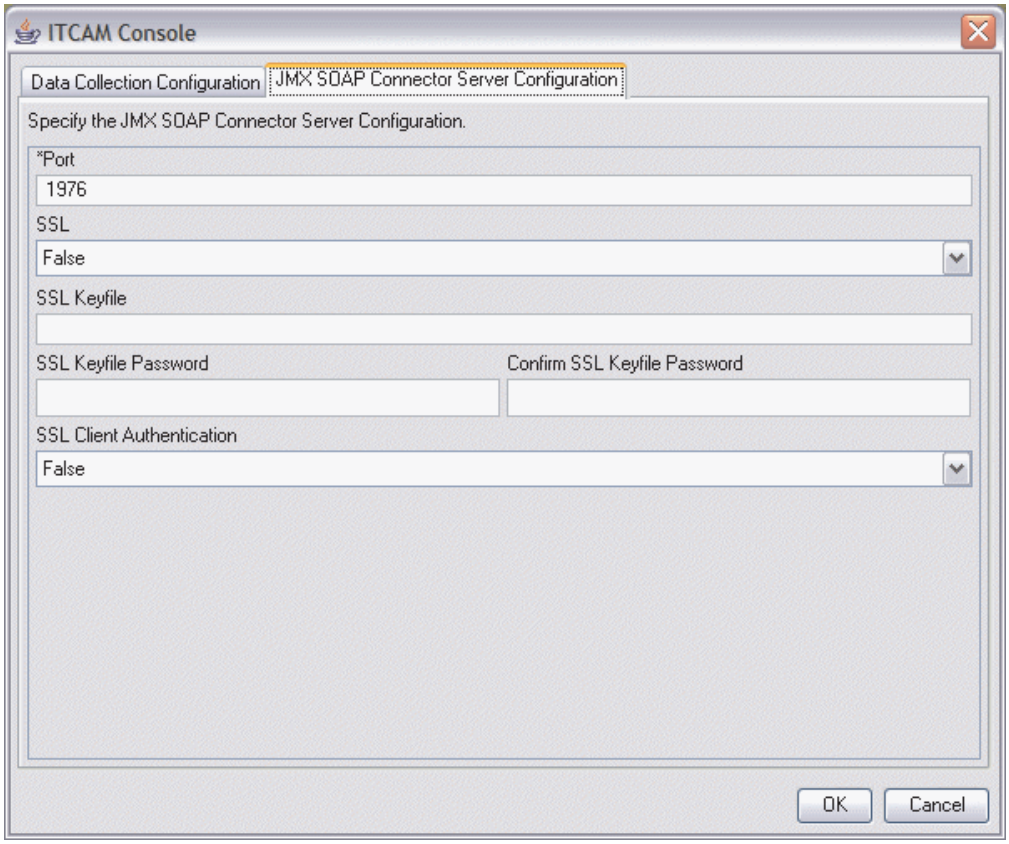
   You can configure the collection frequency by changing the value in the **Data Collection Frequency** field, for example, to collect data at less frequent rates, such as every 10 or 15 minutes. Setting a lower collection frequency is useful to decrease the number of calls made through your monitoring server in high volume environments. Note that the Transaction Reporter frequency is fixed and is not affected by this setting.

   The type of data collected can be controlled using the additional agent specific configuration fields. For example, if you do not want to collect any data, and only need the Application Management Console for configuration purposes, you can set the **Exclude data collection from all agents** parameter to **True**.

   You can filter out specific agents by setting one or more of the following configuration parameters to *True*:

   - **Exclude Client Response Time agent data collection**
   - **Exclude Internet Service Monitor agent data collection**
   - **Exclude Robotic Response Time agent data collection**
   - **Exclude Transaction Tracking agent data collection**
   - **Exclude Web Response Time agent data collection**

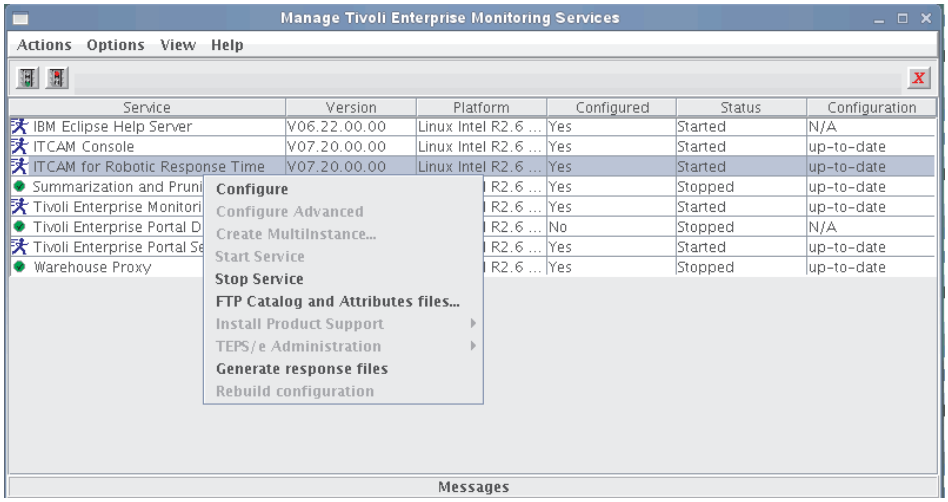
7. Select the **JMX SOAP Connector Server Configuration** tab.



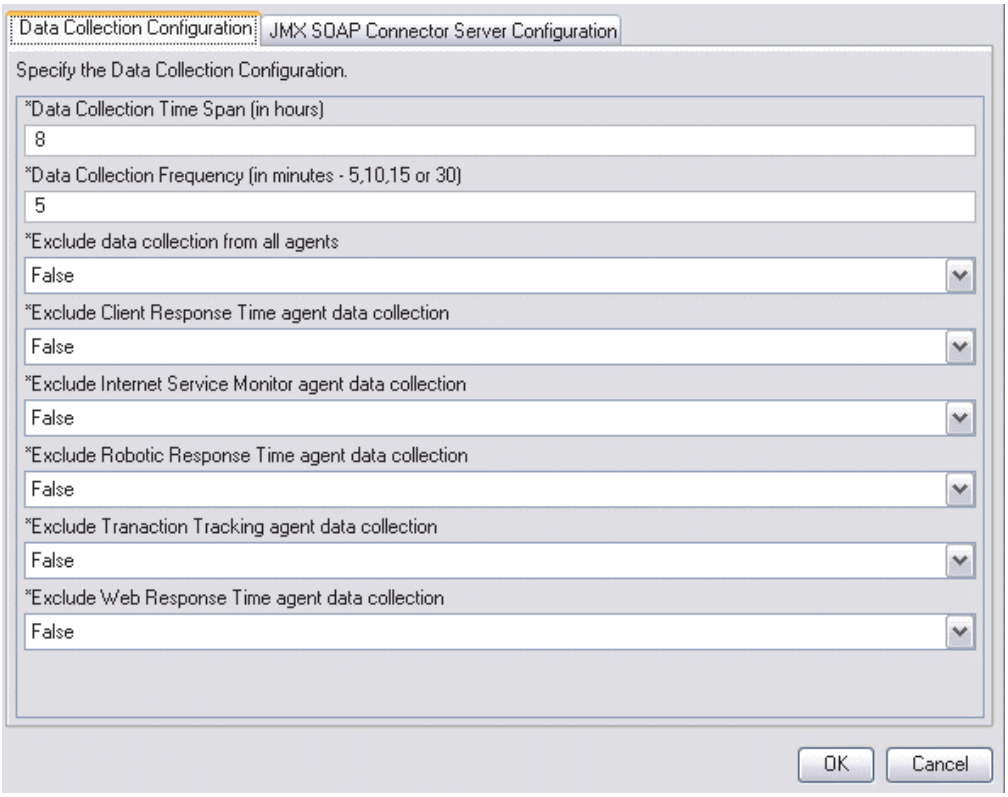
8. Specify the JMX SOAP Connector Server configurations.
9. *(Optional):* When installation is complete, the InstallShield Wizard Complete window opens. Clear the **Display the README File** check box if you do not want to view the readme file.
10. Click **Finish**.
11. Once the installation is complete, proceed to "Verify installations of Response Time monitoring agents" on page 125.

Configuring Application Management Console from the GUI (UNIX and Linux)

1. Make sure that you understand and have collected the configuration parameters for this agent from "Application Management Console parameters" on page 178 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. Open a command prompt and navigate to the `<install_dir>/bin` directory, where `<install_dir>` is the installation location for the monitoring agent. The default installation location is `/opt/IBM/ITM`.
3. Run the **`./itmcmd manage`** command to display the Manage Tivoli Enterprise Monitoring Services window.



4. Right click the agent that you want to configure.
5. In the pop-up menu, click **Configure**.
6. Click the **Data Collection Configuration** tab.



7. Specify the configuration information about how data is analyzed.

   The Application Management Console agent collects its data from other ITCAM for Transactions agents on a regular basis. It does this data collection automatically with no configuration necessary by utilizing the KT1 File Transfer Enablement service installed on the TEMS.

   By default, the data is collected from the agents every 5 minutes except for the Transaction Reporter, from which the data is collected every 2 minutes to

ensure not missing any data. After the data is collected, the metrics in the Application Management Console are updated with the latest values.

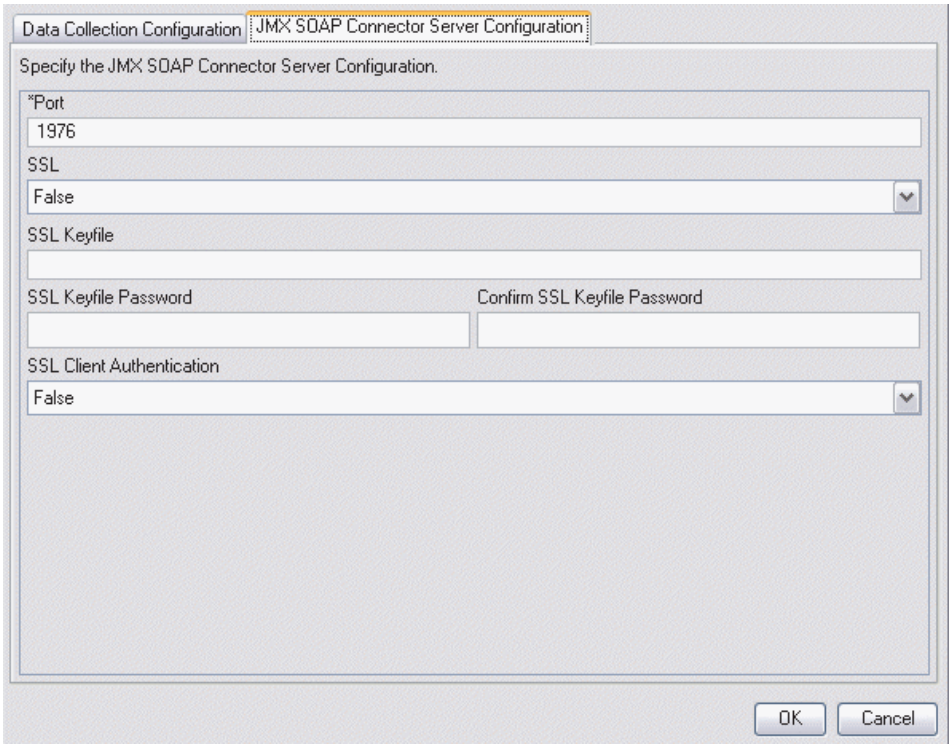
You can configure the collection frequency by changing the value in the **Data Collection Frequency** field, for example to collect data at less frequent rates, such as every 10 or 15 minutes. Setting a lower collection frequency is useful to decrease the number of calls made through your TEMS in high volume environments. Note that the Transaction Reporter frequency is fixed and is not affected by this setting.

The type of data collected can be controlled using the additional agent specific configuration fields. For example, if you do not want to collect any data, and only need the Application Management Console for configuration purposes, you can set the **Exclude data collection from all agents** parameter to *True*.

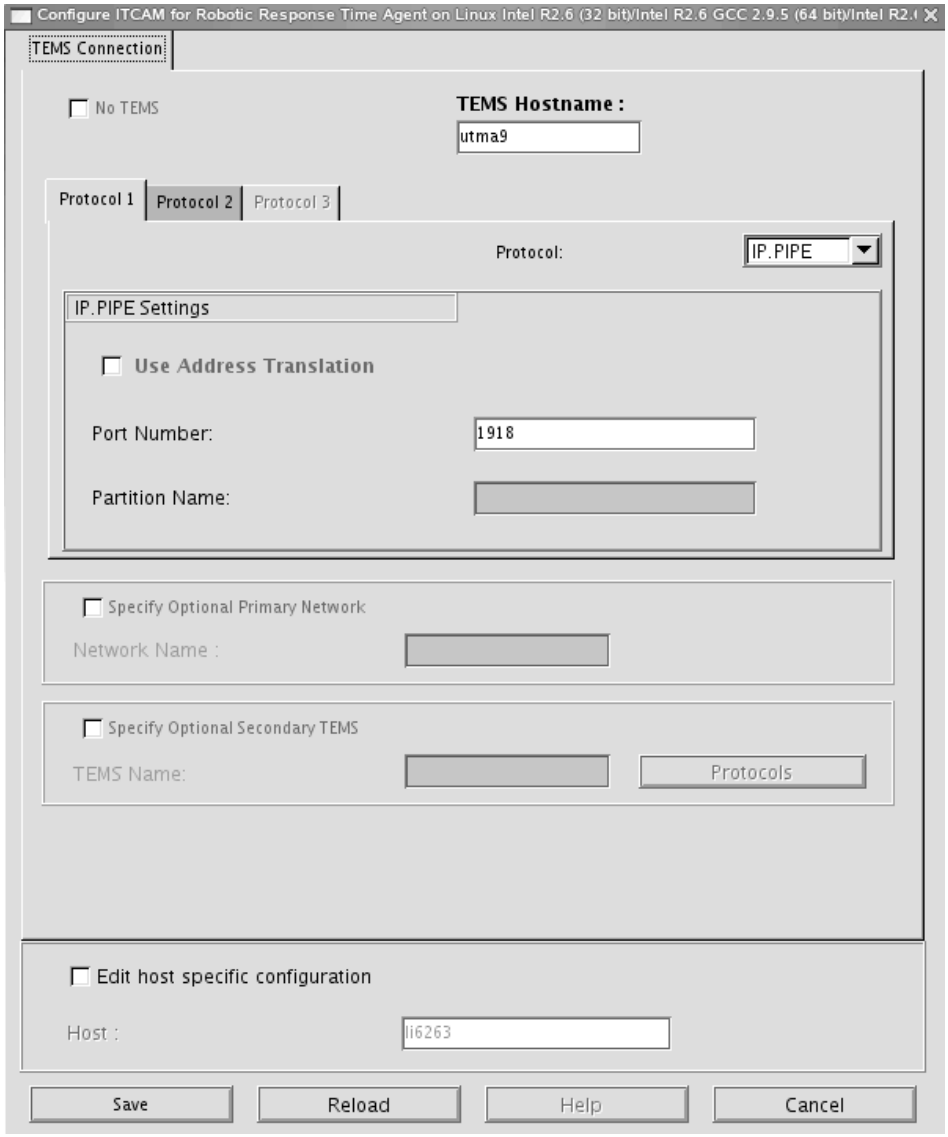
You can filter out specific agents by setting one or more of the following configuration parameters to *True*:

- **Exclude Client Response Time agent data collection**
- **Exclude Internet Service Monitor agent data collection**
- **Exclude Robotic Response Time agent data collection**
- **Exclude Transaction Tracking agent data collection**
- **Exclude Web Response Time agent data collection**

8. Click the **JMX SOAP Connector Server Configuration** tab.



9. Specify the parameters for JMX SOAP Connector Server Configuration tab.
10. Specify the Tivoli Enterprise Monitoring Server connection settings.



11. Click **Save**.
12. Close the Manage Tivoli Enterprise Monitoring Services console.

Configuring Client Response Time

Administrators can configure Response Time to meet the specific requirements of their environment.

This chapter includes the following topics:

- "Configuring Client Response Time (Windows)" on page 150
- "Configuring Client Response Time from the GUI (UNIX and Linux)" on page 152

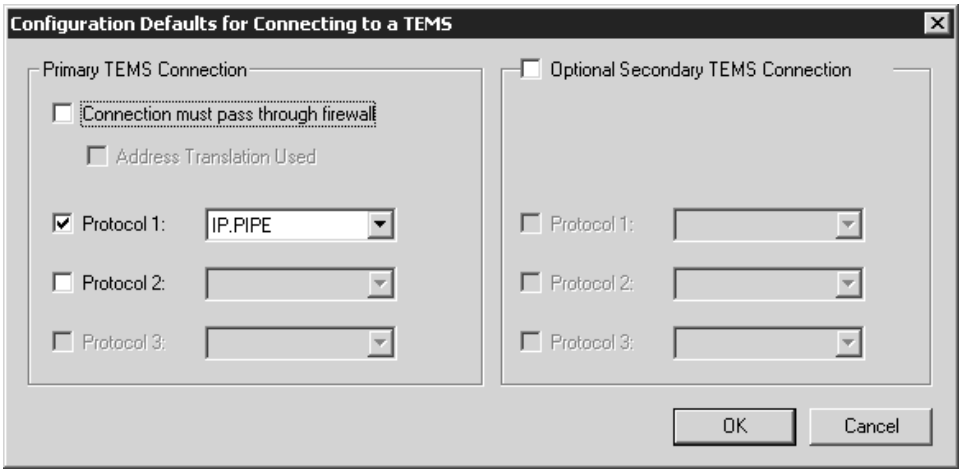
Note: In addition to these methods, you can also configure monitoring agents from the command line. See "Configuring Response Time from the command line (UNIX and Linux) and agent configuration parameters" on page 177.

Configuring Client Response Time (Windows)

Administrators must configure Client Response Time for Windows.

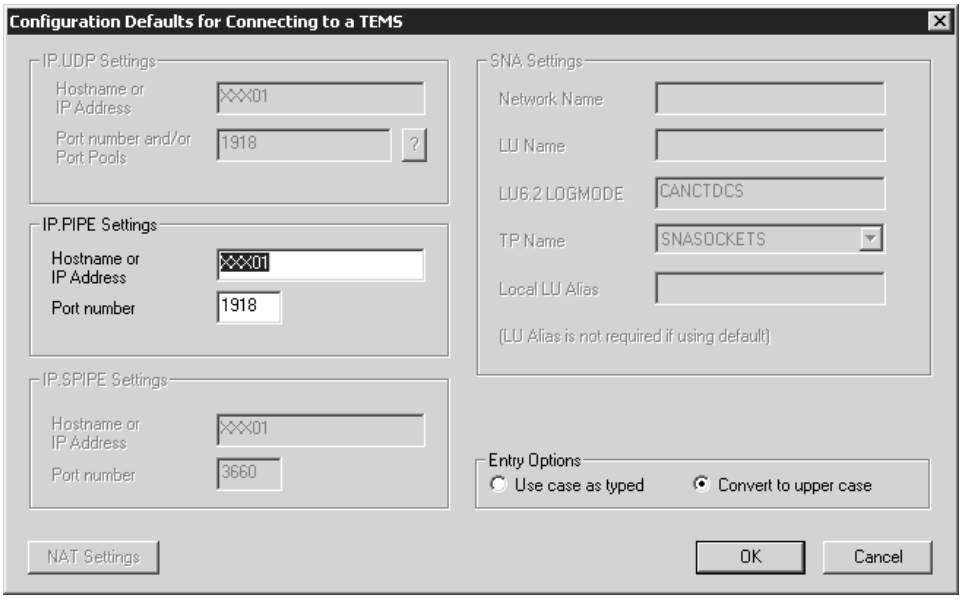
The following procedure tells how to perform a basic configuration:

1. Make sure that you understand and have collected the configuration parameters for this agent from "Client Response Time parameters" on page 178 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. If you selected **Next** from step 16 (in the section called "Install a monitoring agent on Windows systems" on page 102) the software displays the Primary TEMS Connection window.

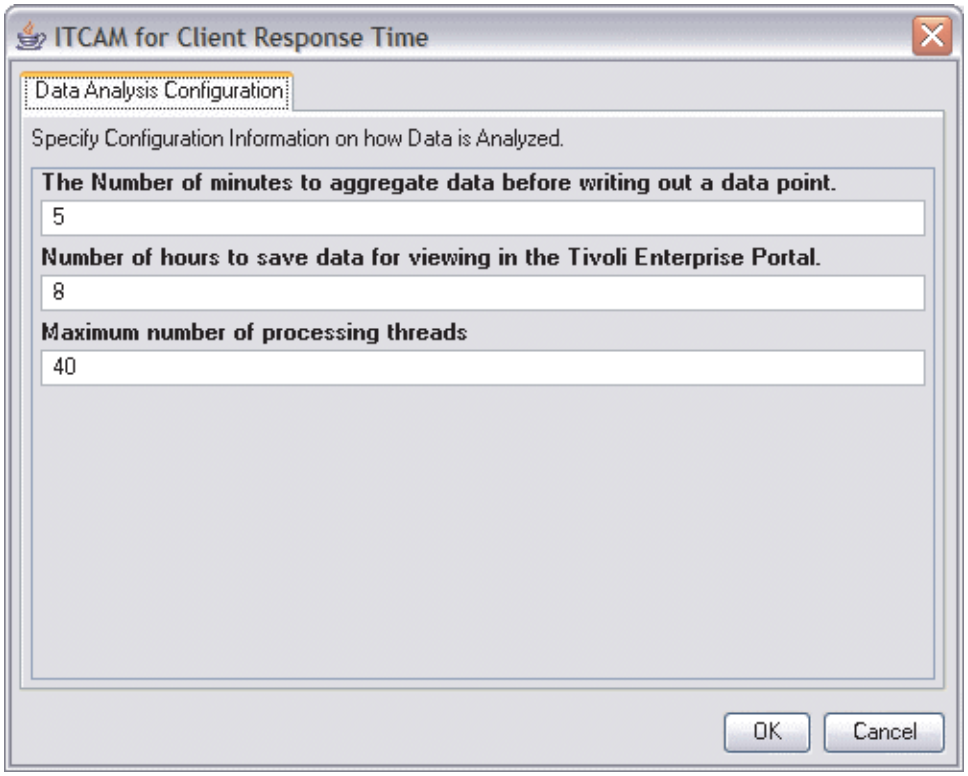


3. Follow these steps to specify the default monitoring server and click **OK**.
   a. If the agent accesses the monitoring server through a firewall, select **Connection must pass through firewall**.
   b. Identify the protocol that the agent uses to communicate with the monitoring server. You have four choices: `IP.UDP`, `IP.PIPE`, `IP.SPIPE`, or `SNA`.

      Note: You can specify up to three communication methods. This enables you to set up backup communication methods. If the method you have identified as Protocol 1 fails, Protocol 2 is used.
   c. Do not select **Optional Secondary TEMS Connection**. You can set up the standby support for agents later. See the IBM Tivoli Monitoring product documentation.



4. Define the communication protocols between agents and the monitoring server. If you are certain that you have typed the values for all of these fields with exactly the correct casing (upper and lower cases), you can select **Use case as typed**. However, because IBM Tivoli Monitoring is case-sensitive, consider selecting **Convert to upper case** to reduce the chance of user error. Click **OK** to display the Configuration window.
5. Specify the configuration information about how data is analyzed and click **OK**.



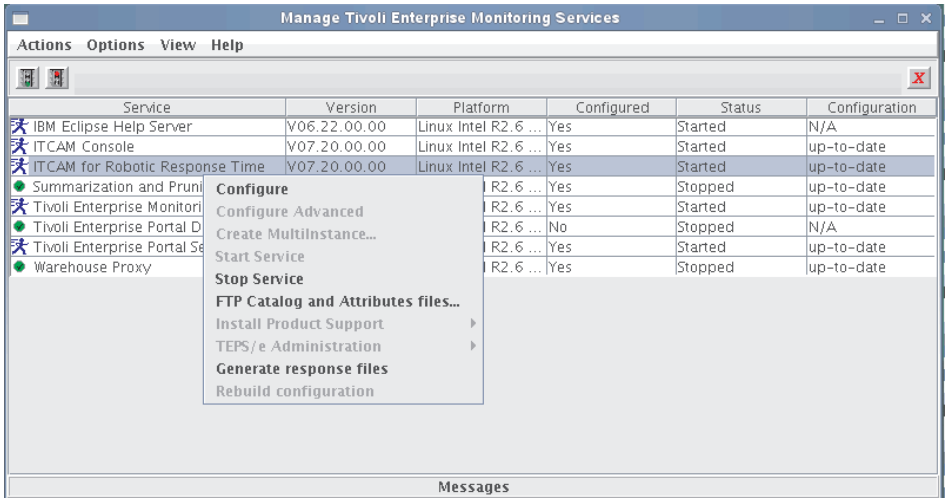
6. Clear the **Display the README File** check box if you do not want to view the readme file.

7. Click **Finish** to complete the installation.
8. To enable generic ARM 2.0 and generic ARM 4.0, complete the following steps:
   a. Make sure the environment variable `CANDLE_HOME` is set and points to the installation directory (default is `C:\IBM\ITM`).
   b. Ensure that the environment variable `PATH` includes the directory where `libarm4.dll` and `libarm32.dll` are located; this is typically the installation directory.
   c. Set the Java classpath when running a Java program to where the `armjni*.jar` files are located.
9. Once the installation is complete, proceed to "Verify installations of Response Time monitoring agents" on page 125.

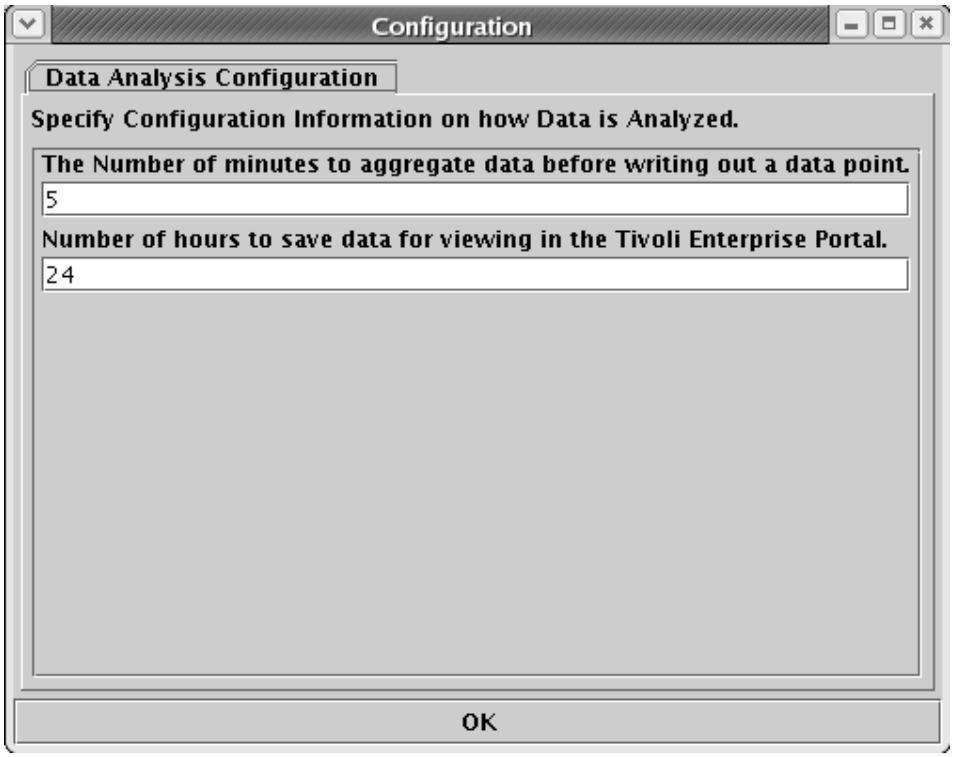
Configuring Client Response Time from the GUI (UNIX and Linux)

Administrators can configure Client Response Time for UNIX and Linux.

1. Make sure that you understand and have collected the configuration parameters for this agent from "Client Response Time parameters" on page 178 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. Open a command prompt and navigate to the `<install_dir>/bin` directory.
   where `<install_dir>` is the installation location for the monitoring agent. The default installation location is `/opt/IBM/ITM`.
3. Run the **`./itmcmd manage`** command to display the Manage Tivoli Enterprise Monitoring Services window.



4. Right-click the agent that you want to configure.
5. In the pop-up menu, click **Configure**.
6. Specify the Data Analysis Configuration.



7. Click **OK** to display the **TEMS Connection** window.
8. Specify the TEMS connection settings and click **Save**.
9. Close the Manage Tivoli Enterprise Monitoring Services window.
10. To enable generic ARM 2.0 and generic ARM 4.0, do the following.
    a. Make sure the environment variable `CANDLE_HOME` is set and points to the IBM Tivoli Monitoring install directory (default is `/opt/IBM/ITM`).
    b. Run the following command.

```
#export CANDLE_HOME=<install_dir>/tmaitm6/
```

Where <*install_dir*> is the installation location for the monitoring agent.

c. Set the library path:

Platform	Library path
AIX	`#export LIBPATH=$CANDLE_HOME/`*`platform`*`/lib`
Solaris, UNIX, or Linux	`#export LD_LIBRARY_PATH=$CANDLE_HOME/`*`platform`*`/lib`
HPUX	`#export SHLIB_PATH=$CANDLE_HOME/`*`platform`*`/lib`

In the table, *platform* is the platform name. For example, `aix523`, `li6263`, `sol283`, `hp11`.

d. When running a Java program, set the Java classpath to where the `armjni*.jar` files are located.

Configuring Robotic Response Time

Administrators can configure Robotic Response Time to meet the specific requirements of their environment.

Configuration topics include:

- "Configuring Robotic Response Time (Windows)"
- "Configuring Robotic Response Time from the GUI (UNIX and Linux)" on page 160

Note: In addition to these methods, you can also configure monitoring agents from the command line. See "Configuring Response Time from the command line (UNIX and Linux) and agent configuration parameters" on page 177

Tip: Rational Robot 7 sometimes becomes unresponsive. To prevent this problem, set the **`Reboot When Robot Not Responding`** parameter to `true`, and enter *Username* and *Password* in the Rational Robot GUI Configuration.

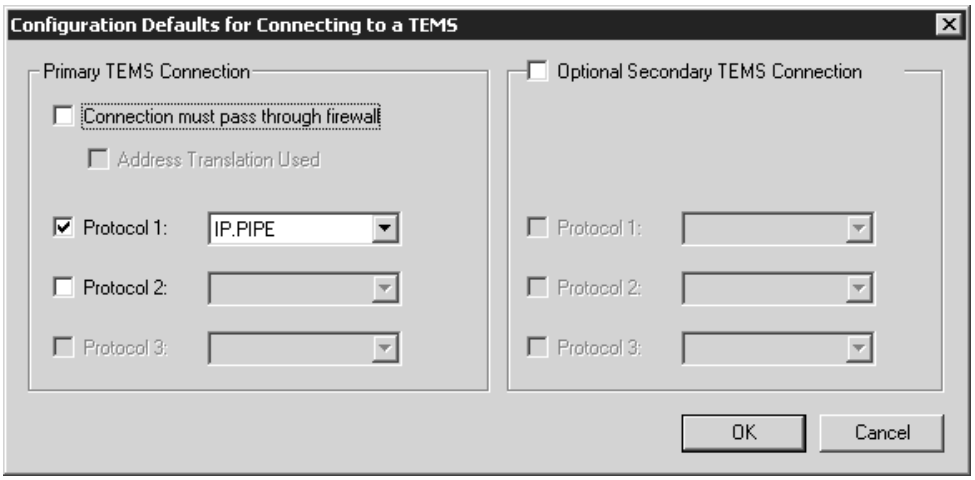
Configuring Robotic Response Time (Windows)

Administrators can configure Robotic Response Time for Windows.

Configuring Robotic Response Time provides the parameter definitions for playing back robotic scripts. You can override the default parameter values within each **configuration tab**, but if you do not change the values, the installation uses the default parameter values.

1. Make sure that you understand and have collected the configuration parameters for this agent from"Robotic Response Time parameters" on page 179 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. If you selected **Next** from step 16 (in the section called "Install a monitoring agent on Windows systems" on page 102) the installation software displays the Tivoli Enterprise Monitoring Server Configuration window.
3. This window displays the default responses based on the TEMS configuration set up for IBM Tivoli Monitoring. Click **OK** to display the Hub TEMS Configuration window.

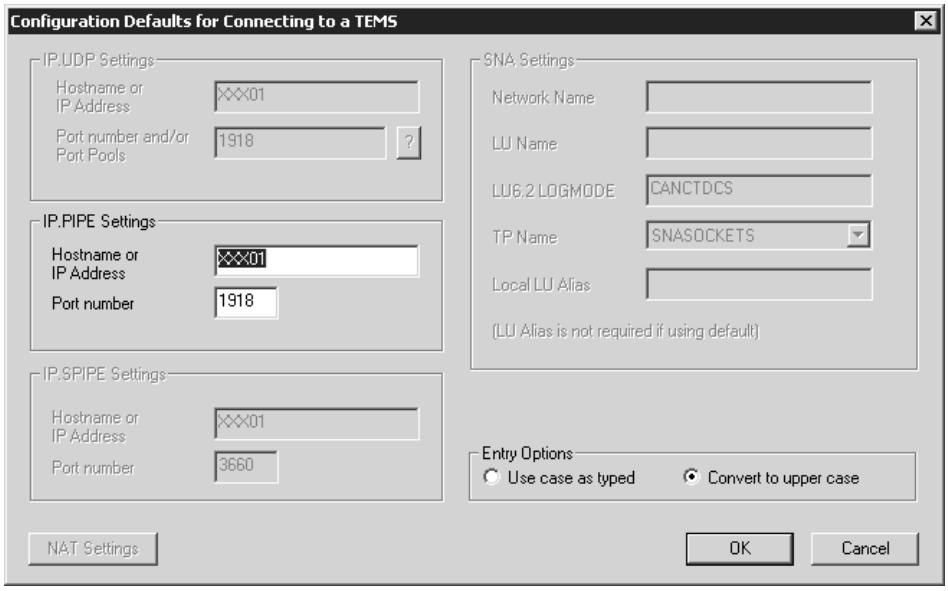
4. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:



   a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.
   b. Identify the Information to collect before you begin installation and configuration that the agent uses to communicate with the monitoring server. You have four choices: `IP.UDP`, `IP.PIPE`, `IP.SPIPE`, or `SNA`.

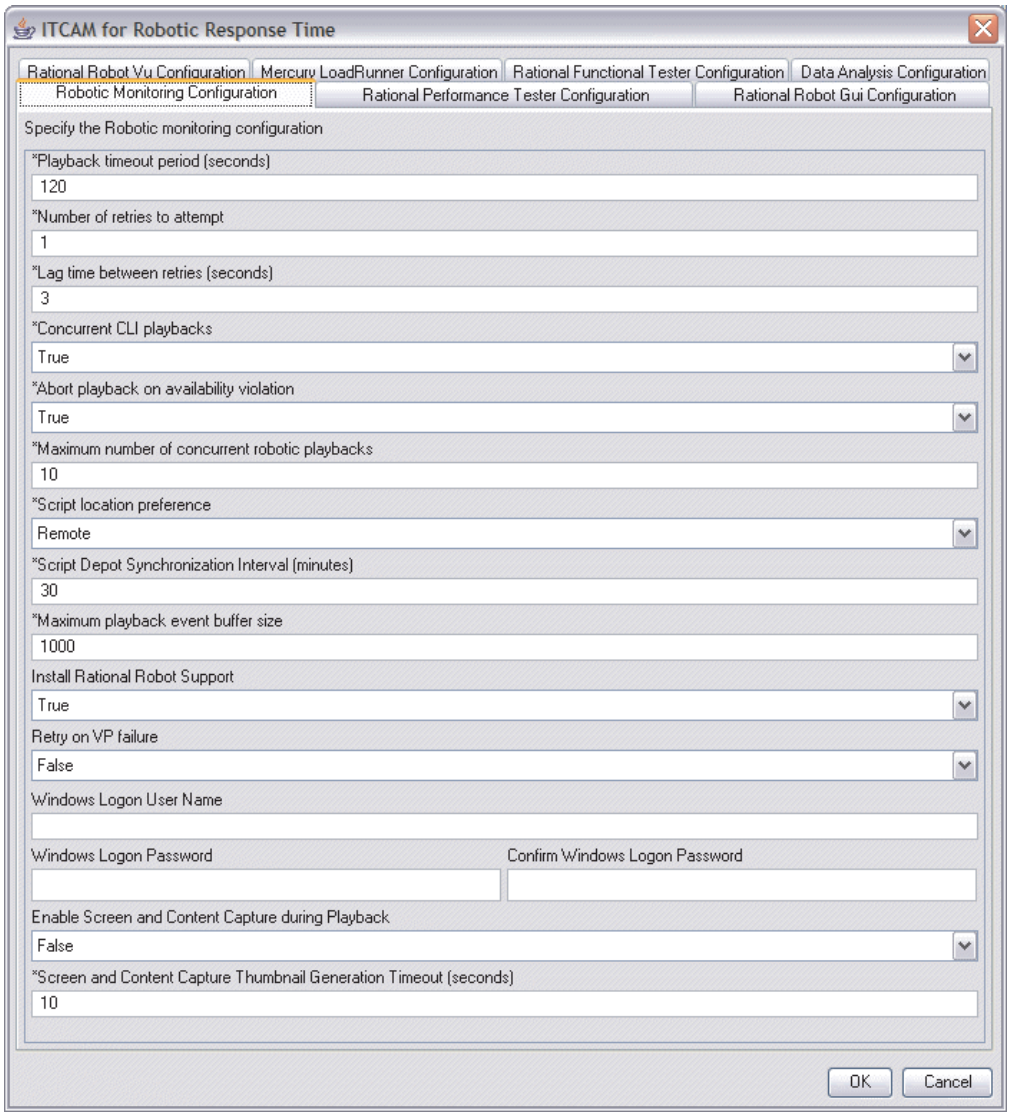
      Note: You can specify three communication methods. This enables you to set up backup communication methods. If the method you have identified as Protocol 1 fails, Protocol 2 is used.
   c. Do not select **Optional Secondary TEMS Connection**. You can set up the standby support for agents after installation. See the IBM Tivoli Monitoring product documentation.

5. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**. The information on this window is automatically detected and depends on the protocol selected earlier.

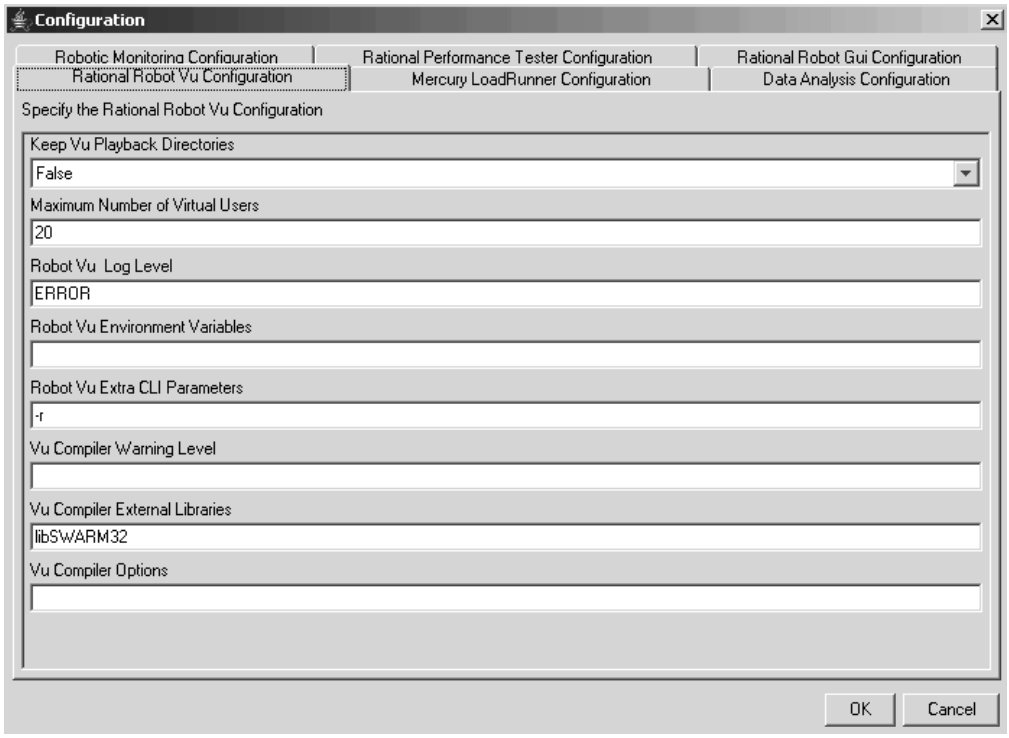


6. In the Configuration window, do one of the following:

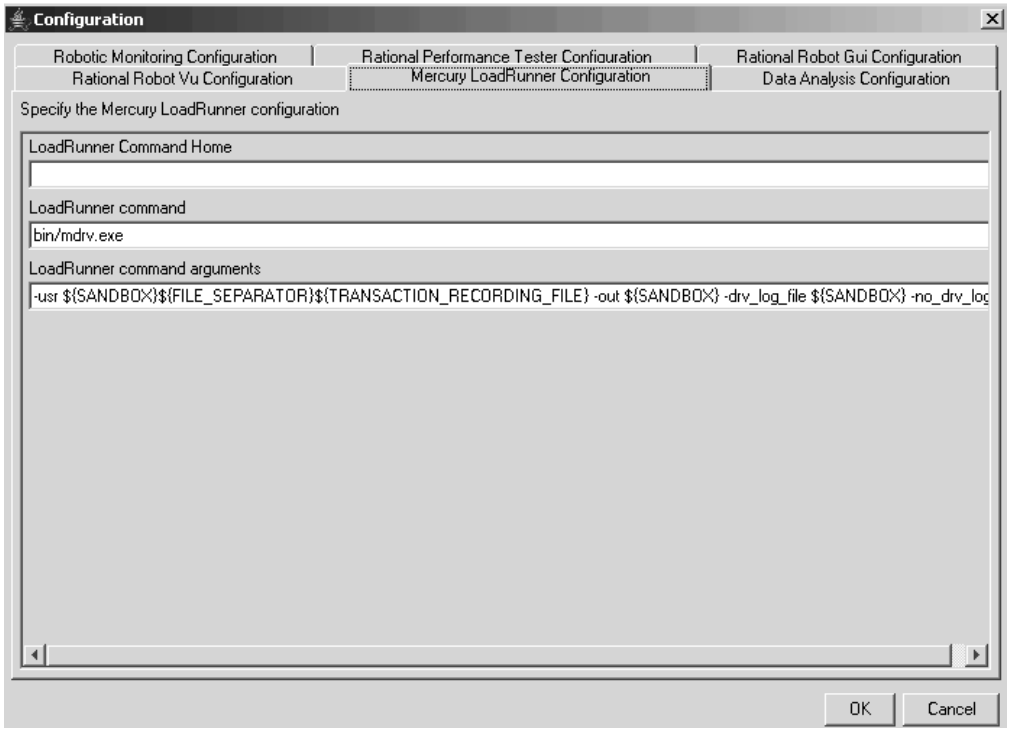
- Accept all the default parameter values by clicking **OK**.
- Customize the parameter values in the Configuration tabs:
  - Select the **Robotic Monitoring Configuration** tab.



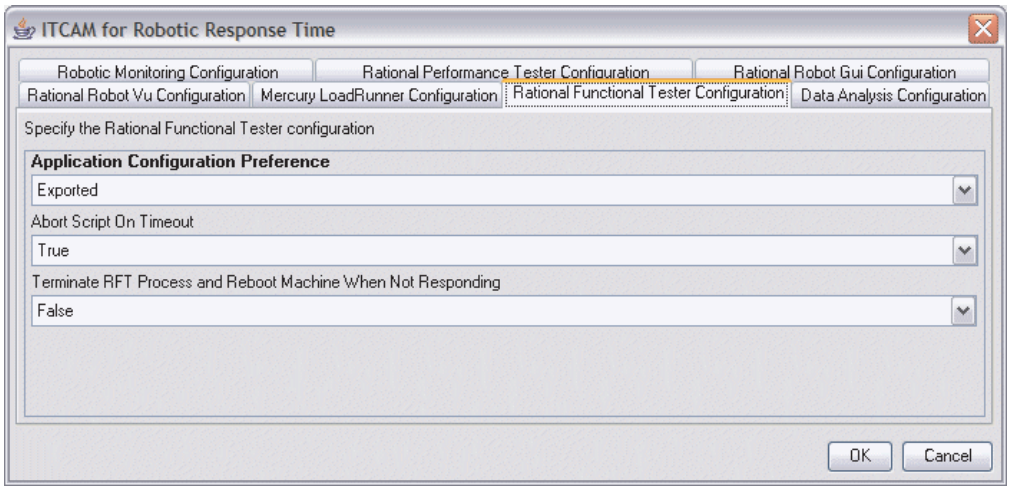
  - Select the **Rational Robot VU Configuration** tab.



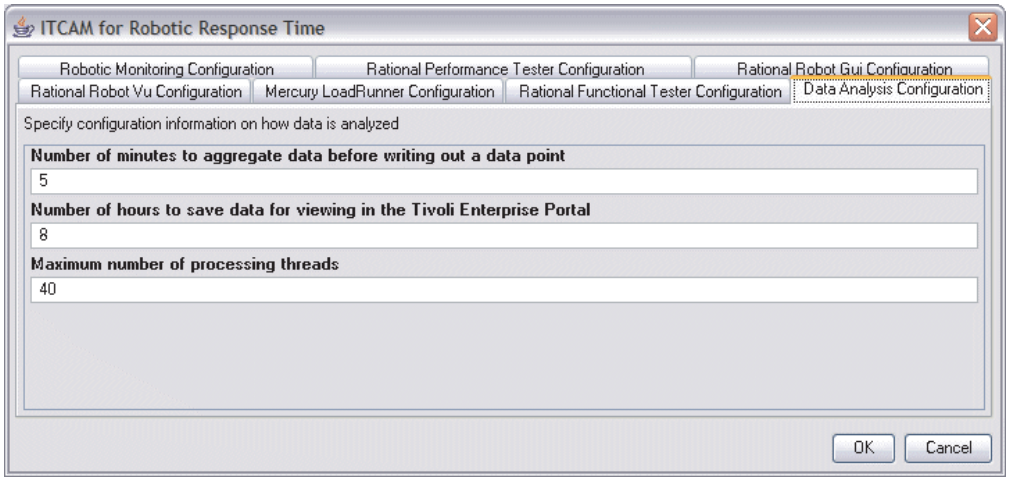
– Select the **Mercury LoadRunner Configuration** tab and use the information from "Robotic Response Time parameters" on page 179.



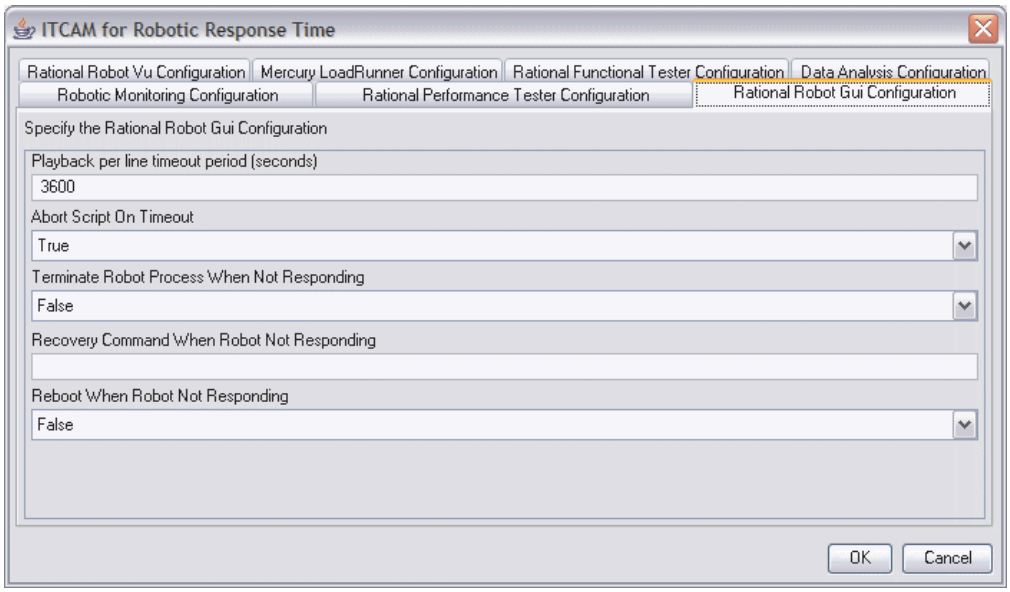
– Select the **Rational Functional Tester Configuration** tab.



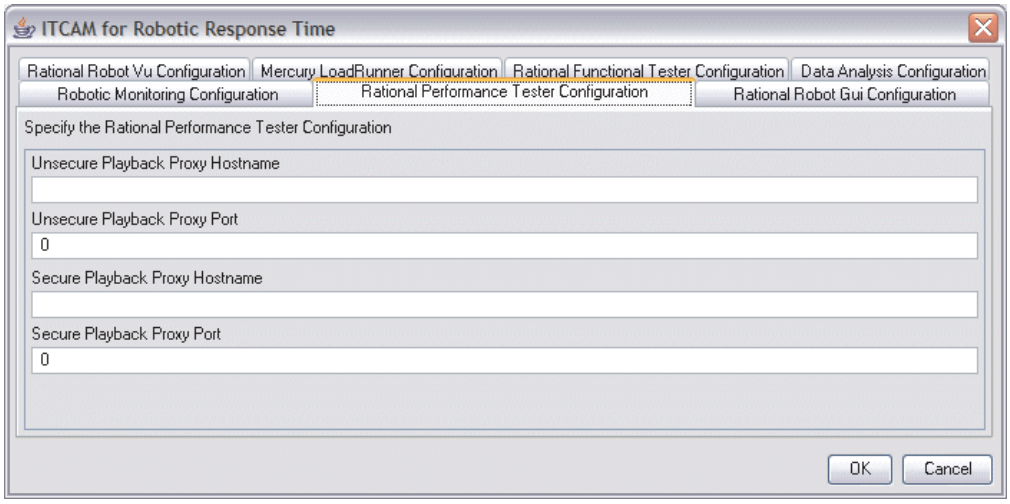
– Select the **Data Analysis Configuration** tab.



– Select the **Rational Robot Gui Configuration** tab.



– Select the **Rational Performance Tester Configuration** tab.

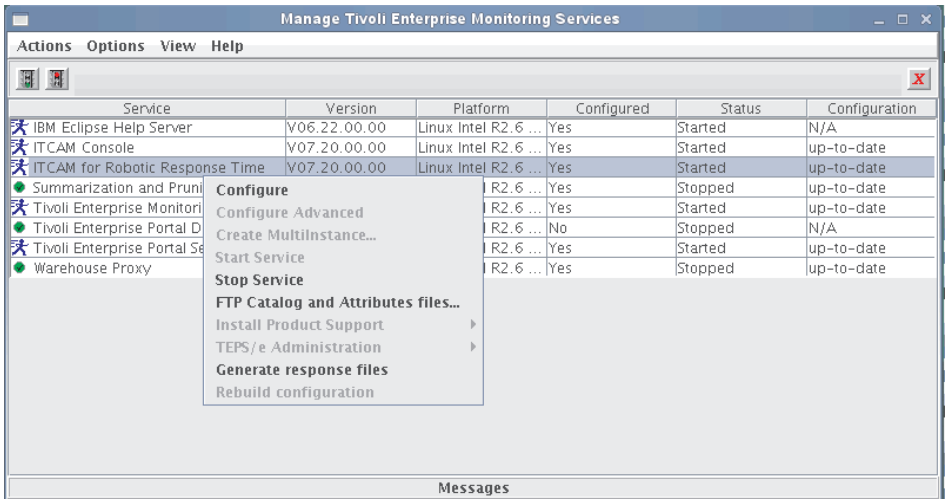


7. *(Optional):* When installation is complete, the InstallShield Wizard Complete window opens. Clear the **Display the README File** check box if you do not want to view the readme file.
8. Click **Finish**.

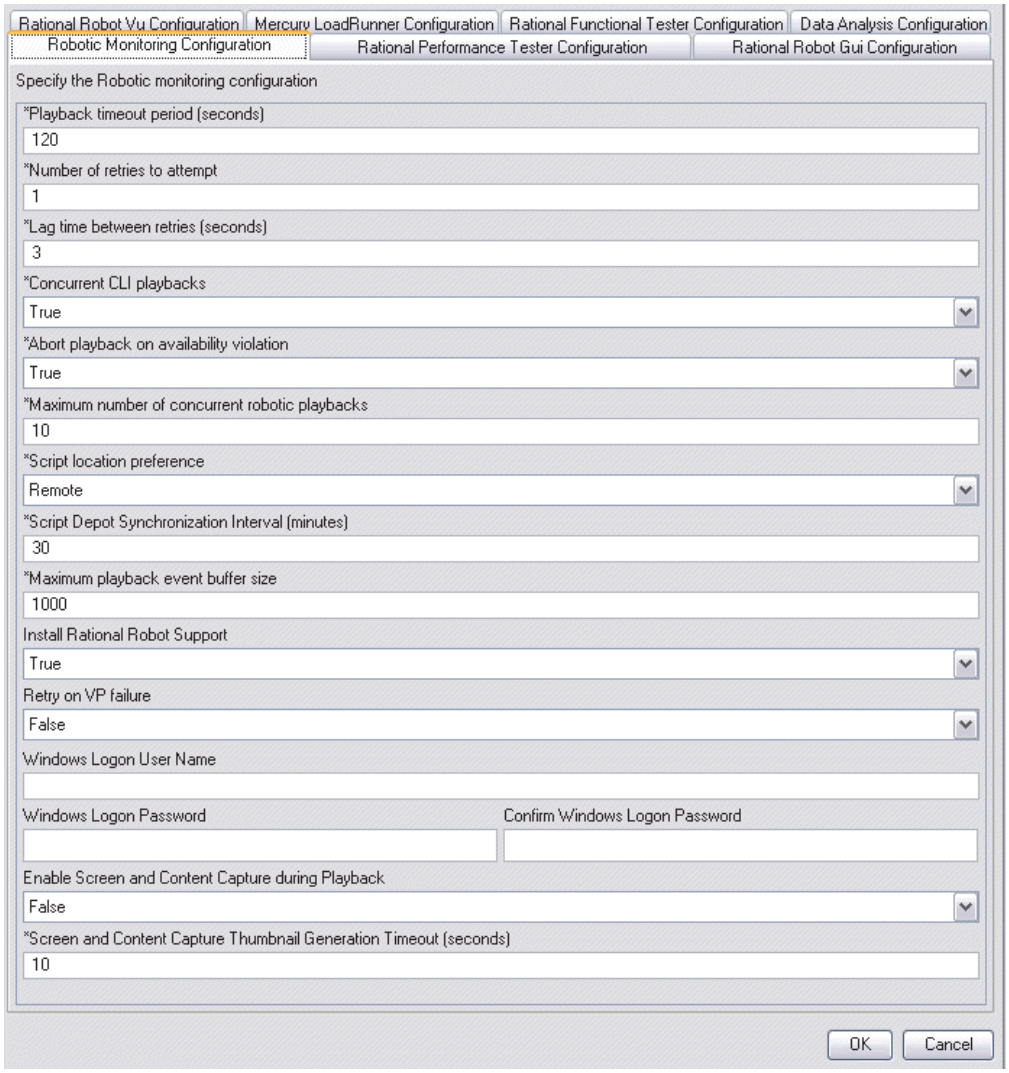
   Note: If you remotely install and configure this agent and you are monitoring a local HTTPS server using the web server plugin, restart the HTTP server.
9. Robotic Response Time requires a reboot after the installation is complete. Reboot your environment before verifying the installation.
10. Once the installation is complete, proceed to “Verify installations of Response Time monitoring agents” on page 125.

Configuring Robotic Response Time from the GUI (UNIX and Linux)

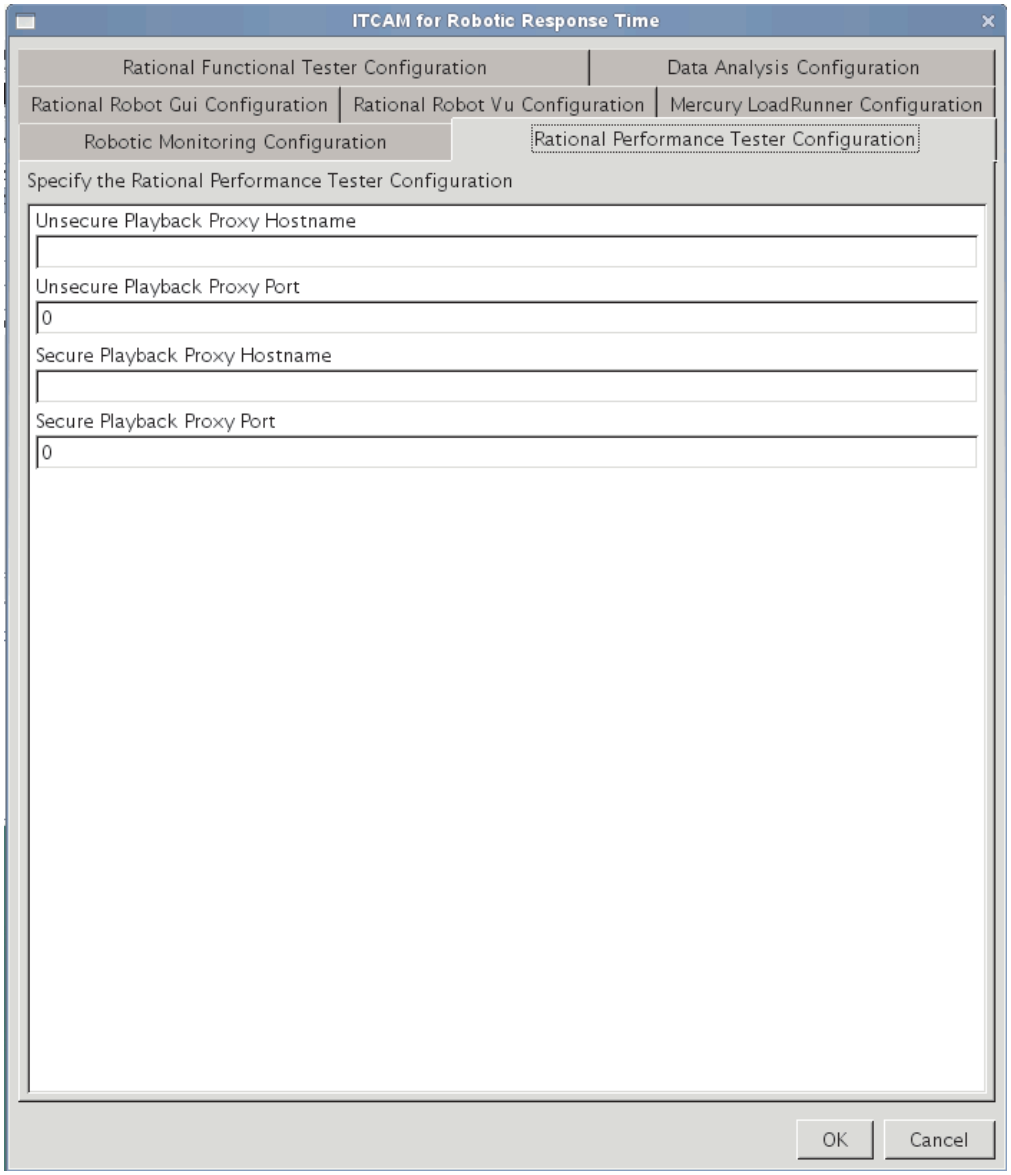
1. Make sure that you understand and have collected the configuration parameters for this agent from"Robotic Response Time parameters" on page 179 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. Open a command prompt and navigate to the <*install_dir*>/bin directory, where <*install_dir*> is the installation location for the monitoring agent. The default installation location is /opt/IBM/ITM.
3. Run the **./itmcmd manage** command to display the Manage Tivoli Enterprise Monitoring Services window.



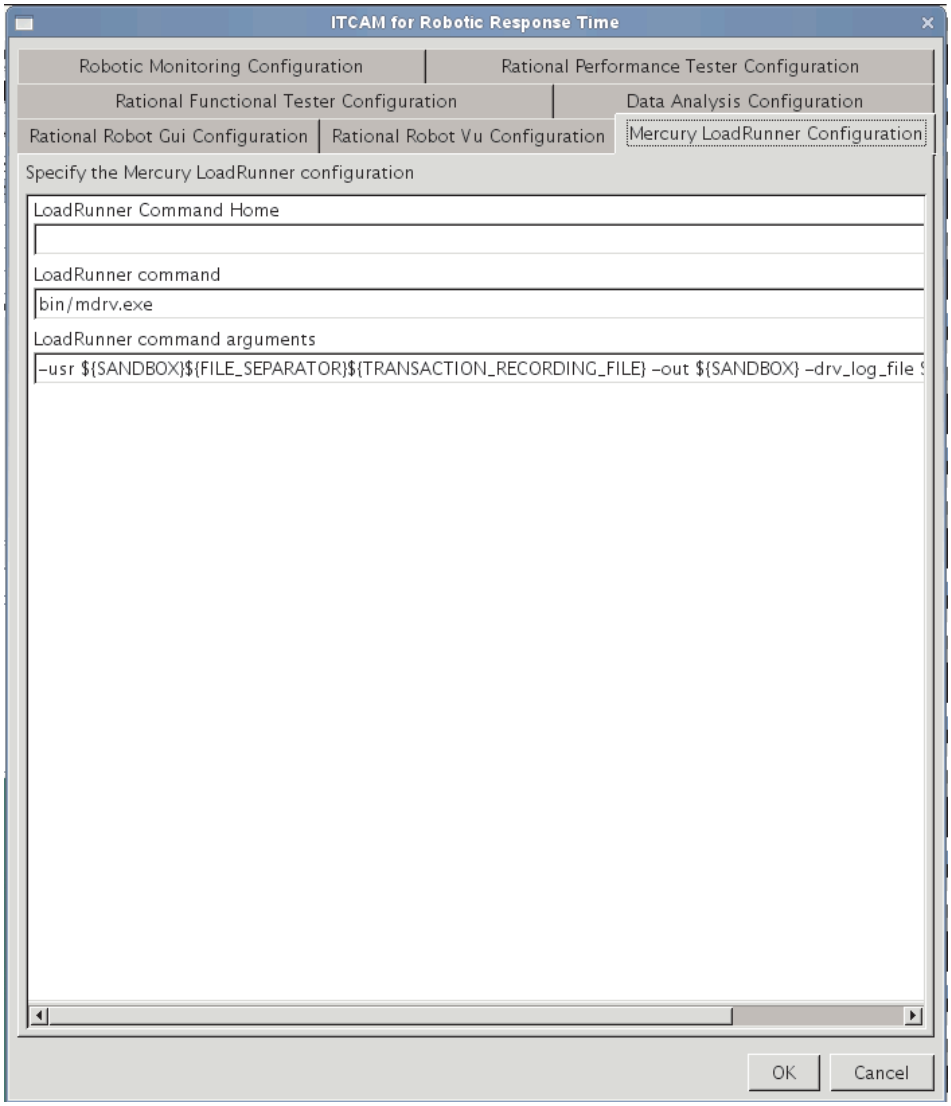
4. Right click the agent that you want to configure, in this case Robotic Response Time.
5. In the pop-up menu, click **Configure**. Another window is displayed, with multiple tabs for configuring the Robotic Response Time agent for various supported environments. Note that for Linux systems, the following tabs are not supported and should be ignored:
   - Rational Robot Gui Configuration
   - Rational Robot VU Configuration
   - Rational Functional Tester Configuration
6. Select the **Robotic Monitoring Configuration** tab and specify the configuration parameters for Robotic Monitoring.



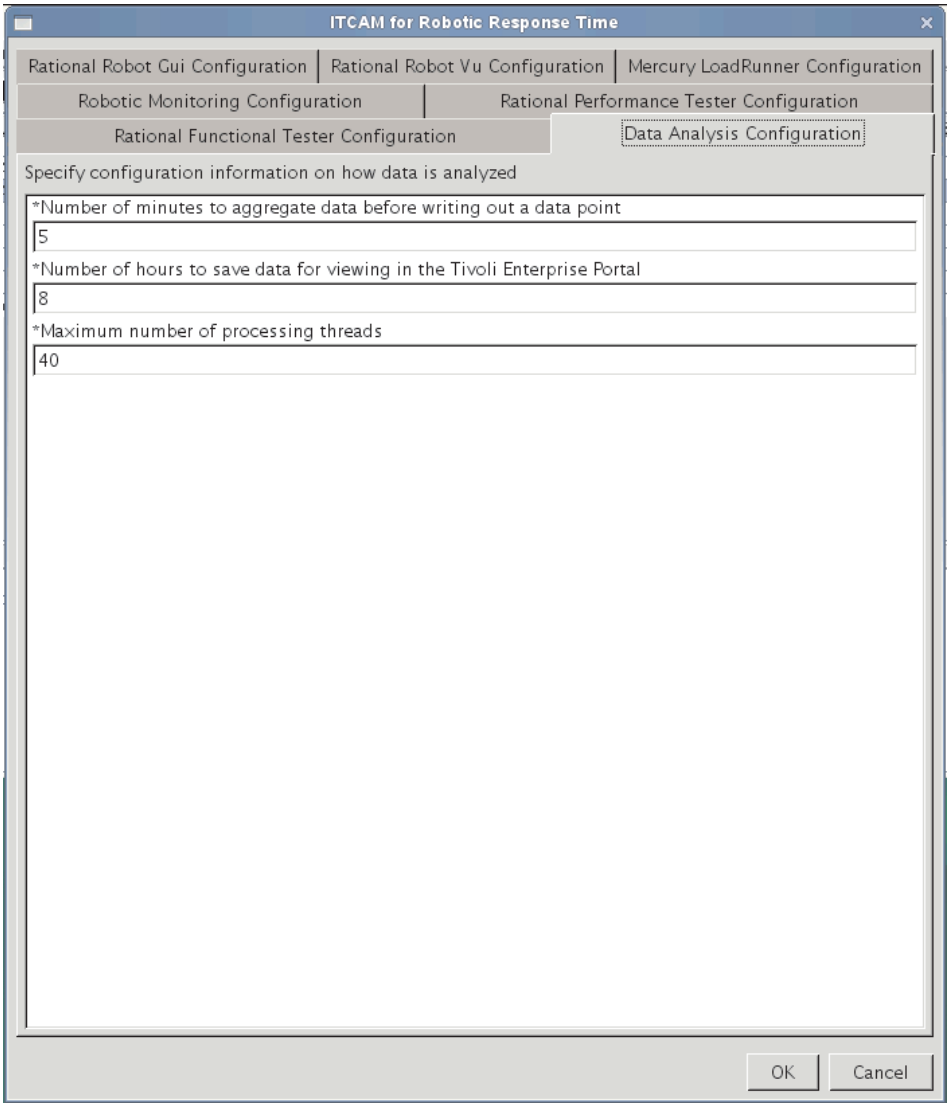
7. Select the **Rational Performance Tester Configuration** tab and specify the configuration parameters for Rational Performance Tester on Linux systems only. See “Configuring Robotic Response Time” on page 154 for further information. Rational Performance Tester is not supported on other UNIX systems.



8. Click the **Mercury LoadRunner Configuration** tab and specify the configuration parameters for Mercury LoadRunner.



9. Select the **Data Analysis Configuration** tab and specify the configuration parameters for Data Analysis.



10. Click **OK**. If you are prompted to enter connection details for Tivoli Enterprise Monitoring Server, do so and save your settings.
11. Close the Manage Tivoli Enterprise Monitoring Services window to complete the configuration.

Configuring Web Response Time

Administrators can configure Web Response Time to meet the specific requirements of their environment.

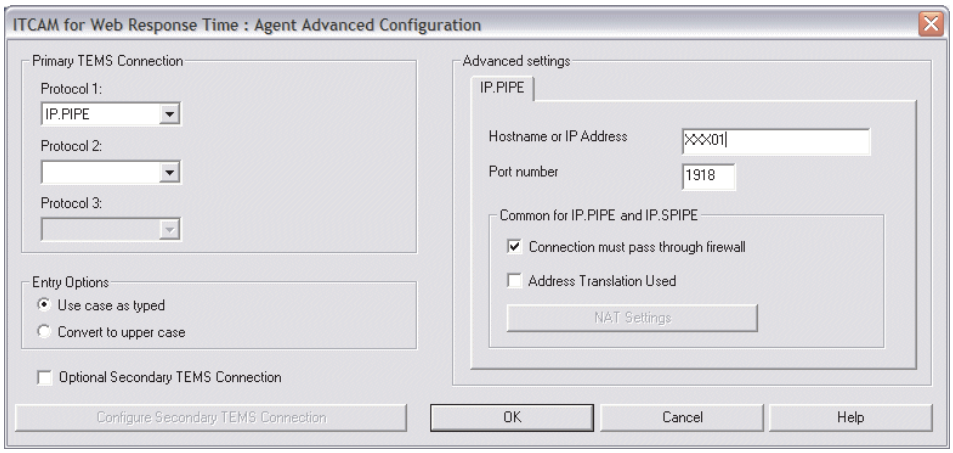
This chapter includes the following topics:

Note: In addition to these methods, you can also configure monitoring agents from the command line. See "Configuring Response Time from the command line (UNIX and Linux) and agent configuration parameters" on page 177.

Configuring Web Response Time (Windows)

Administrators can configure Web Response Time for Windows.

1. Make sure that you understand and have collected the configuration parameters for this agent from "Web Response Time parameters" on page 182 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. If you selected **Next** from step 16 (in the section called "Install a monitoring agent on Windows systems" on page 102), the installation software displays the Agent Advanced Configuration window.
3. Specify the appropriate Tivoli Enterprise Monitoring Server connection information:



   a. In the **Primary TEMS Connection** section, specify one or more connection methods to communicate with the monitoring server. You have four choices: `IP.UDP`, `IP.PIPE`, `IP.SPIPE`, or `SNA`.

      Note: You can specify three communication methods. This enables you to set up backup communication methods. If the method you have identified as Protocol 1 fails, Protocol 2 is used.

   b. In the **Entry Options** section, select the appropriate radio button to control whether entries are accepted as typed or all changed to upper case.

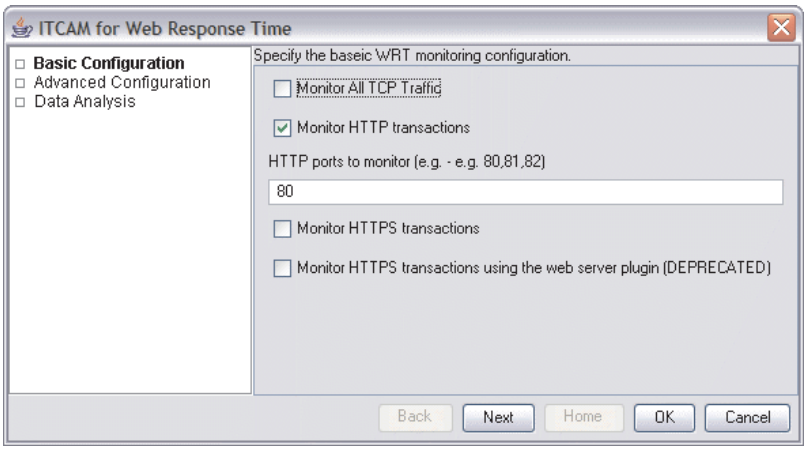
   c. In the Advanced Settings section, you are prompted for different information depending on which protocol you selected in the Primary TEMS Connection section. Verify the default information on host name and port, or modify these values along with other information requested as needed.

   d. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.

   e. Do not select **Optional Secondary TEMS Connection**. You can set up the standby support for agents after installation. See the IBM Tivoli Monitoring product documentation.

   f. Click **OK** to continue with Web Response Time agent basic configuration.

4. In the ITCAM for Web Response Time **Basic Configuration** window, select the appropriate checkboxes as needed and enter additional information as requested. Note that the input fields change depending on which options you select. The option to monitor all HTTP transactions is selected by default.



HTTP vs TCP monitoring: If you select **Monitor All TCP Traffic**, TCP traffic is monitored and response time and bandwidth related metrics for this traffic will be available in the Application, Client, and Server workspaces. Using HTTP monitoring, a web request is merged with its object requests (GIF, CSS, etc), compared to TCP monitoring, in which each of these requests is counted separately. Therefore, a single HTTP transaction can equate to many TCP transactions. So, depending on how you configure your Web Response Time agent, this information is displayed differently in the Web Response Time workspaces from the Network node in the TEP Navigator view. For more information about defining components in the Application Management Configuration Editor, see the *Administrator's Guide*.

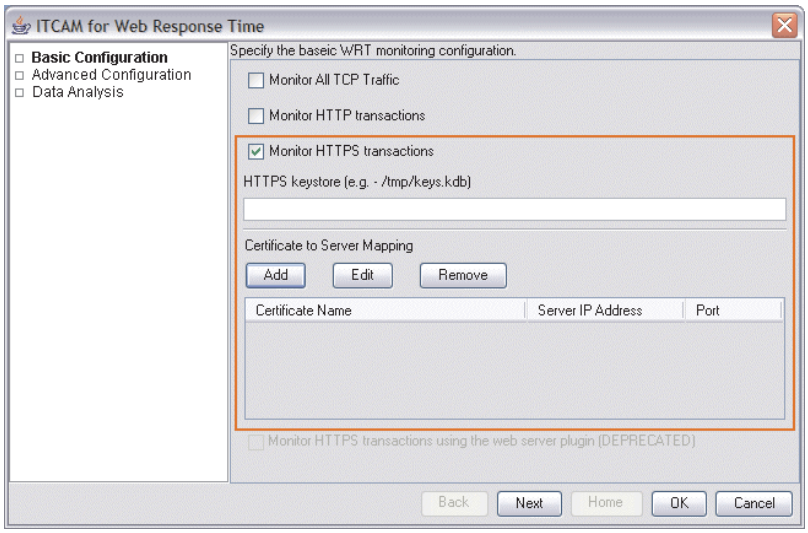
Specify the following basic configuration information:

a. Optionally select **Monitor All TCP Traffic** to enable the agent to monitor TCP traffic.

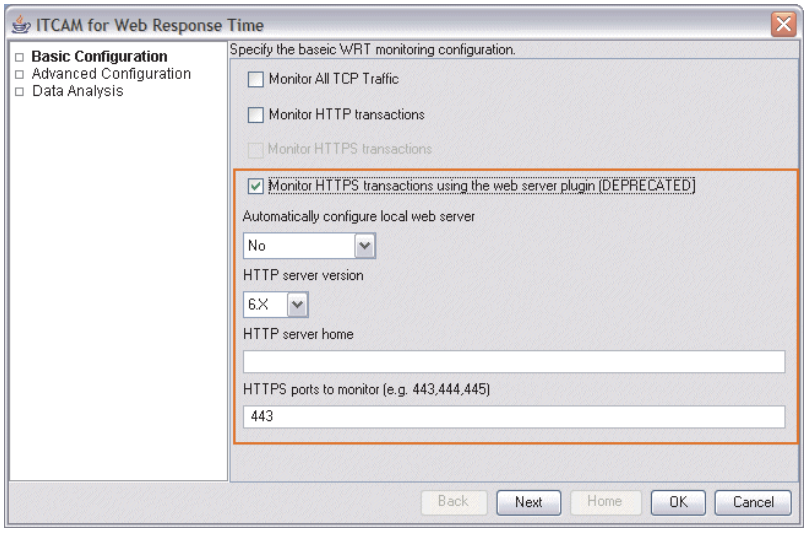
b. Optionally select **Monitor HTTP transactions** to enable the agent to monitor HTTP traffic (this is selected by default). In the **HTTP Ports to Monitor** field, specify one or more ports to monitor, separated by commas, with no blank spaces. The default is port *80*.

c. Optionally select **Monitor HTTPS transactions** to enable the agent to monitor HTTPS traffic. Optionally, specify the following additional information:

   - Specify the HTTPS keystore containing the remote certificates of the remote HTTPS web sites to be monitored.
   - Specify the Certificate to Server Mapping, which defines which remote servers should be monitored, and the SSL certificate used by each remote server. Use the **Add**, **Edit**, and **Remove** buttons to define certificate names, server and port information.



d. Optionally select **Monitor HTTPS transactions using the web server plugin (DEPRECATED)** to enable the agent to monitor HTTPS traffic using a plugin that is installed on the local web server. Optionally, specify the following additional information:
   - Specify whether the monitoring plugin should be automatically configured on the local web server (IBM HTTP Server or Apache).
   - If you specified to automatically configure the monitoring plugin, specify the version of the selected local HTTP server.
   - Specify the home directory of the monitored HTTP server.
   - Specify one or more HTTPS ports to be monitored by the Web Response Time agent, separated by commas, with no blank spaces. The default port is *443*. See Configuring Web Response Time to monitor remote HTTPS transactions in the *Administrator's Guide* for further information.

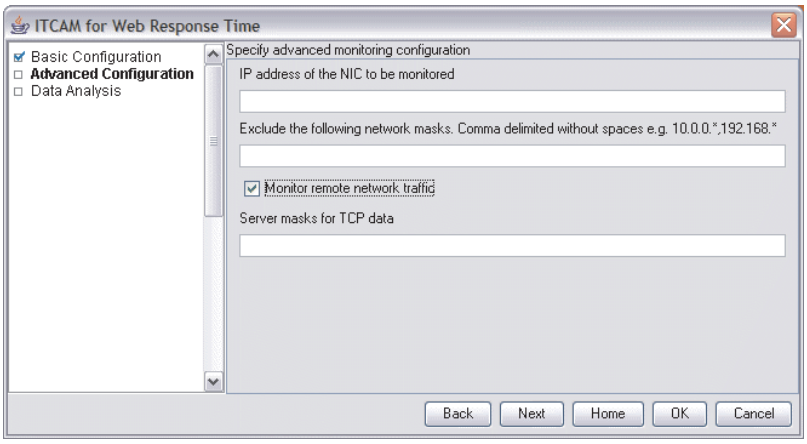


e. Click **Next** to continue to the **Advanced Configuration** window.

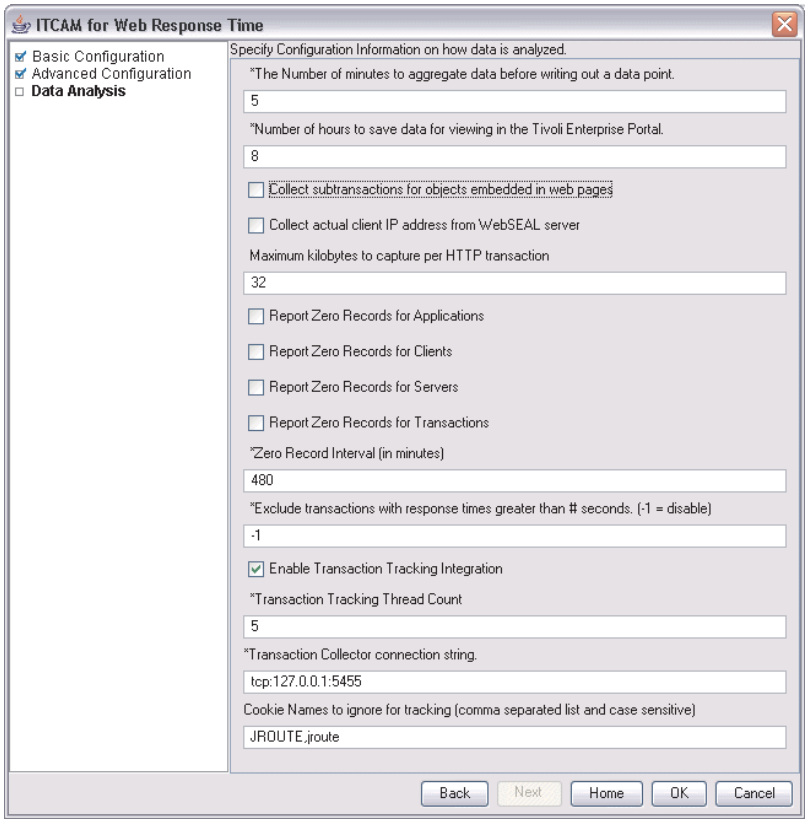
5. In the ITCAM for Web Response Time **Advanced Configuration** window, specify additional optional configuration information as needed.
   - The IP address of the selected NIC card to be monitored.

- One or more network masks to be excluded from monitoring, separated by commas, with no blank spaces. For example, you might enter *9.48.152.*,9.48.164.**, which prevents traffic on subnets 9.48.152 and 9.48.164 from being processed.
- Select the **Monitor remote network traffic** check box if you want to monitor all network traffic on the NIC. If you do not select this option, only the local traffic to and from the local IP address is monitored.
- One or more server masks for TCP data monitoring, separated by commas, with no blank spaces (for example, *10.0.0.*,192.168.**). TCP data coming from an IP address that matches one of these masks is marked as coming from a server, and not a client group.

  Use this field when you are using multiple Web Response Time agents to provide TCP tracking data to Transaction Reporter. In this scenario, to display TCP topology correctly, this data is displayed in the resulting topology view as coming from a server instead of a client. For example, suppose Web Response Time agent *WRT1* observes traffic from clients to server 9.48.152.1, and another Web Response Time agent, *WRT2*, observes traffic from server 9.48.152.1 to server 9.48.152.2. In this case, *WRT2* should be configured with a server mask that includes 9.48.152.1.



6. Click **Next** to continue to the **Data Analysis** dialog and specify additional configuration information defining how the collected data is to be analyzed.



Accept the defaults, or optionally specify the following information:

a. Specify the number of minutes to aggregate data before it is saved and summarized into an aggregate data point. The default is every 5 minutes. For example, to see data in larger aggregate intervals (for SLA reporting without diagnosis), change the default value from 5 minutes to a larger value, such as 15, 30, or 60 minutes, etc

b. Specify the number of hours that data is saved for viewing in the TEP. The default of 8 hours causes the TEP to display aggregate data collected over the last 8 hours.

c. Optionally select the check box to indicate that subtransactions should be collected for embedded objects (GIF, etc) in web pages.

d. Optionally select the check box to indicate that actual client IP addresses should be collected from a WebSEAL server.

e. Specify the maximum number of kilobytes in each HTTP request or reply. The default is 32 kilobytes.

f. Optionally select the check boxes for reporting zero records for applications, clients, servers, or transactions for the duration of the zero record interval.

g. Specify the number of minutes to continue to report zero records. the default is 480 minutes (8 hours).

h. Optionally specify a response time threshold value in seconds, which causes any transactions with response times greater than the threshold value to be excluded from all calculations. The default value is *-1*, which disables this function and includes all transactions, regardless of their response time, in the calculations.

i. Optionally select the **Enable Transaction Tracking Integration** check box to cause the Web Response Time agent to send its data to the Transaction

Collector agent using generated TTAPI events for transaction tracking integration. Accept the additional default values that are displayed, or specify your own values for the following parameters:

- Transaction Tracking Thread Count (default is *5*)
- Transaction Collector connection string
- A comma separated, case-sensitive string of cookie names to ignore during tracking.

Note: You cannot enable transaction tracking integration if HTTPS (plugin mode) is enabled. The Web Response Time agent cannot obtain the necessary tracking information from transactions monitored in HTTPS plugin mode.

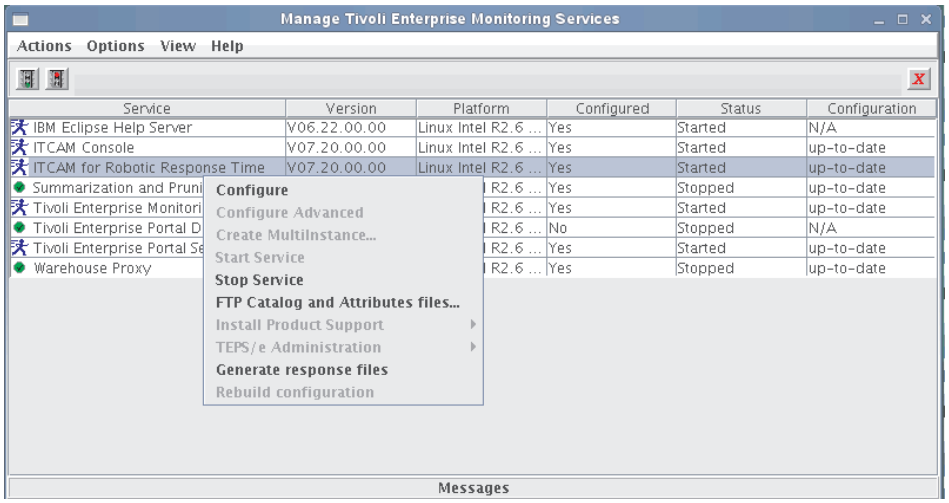
7. Click **OK**. A status window is displayed with the `Configuring and starting agents, please wait` message.
8. *(Optional):* When installation is complete, the InstallShield Wizard Complete window opens. Clear the **Display the README File** check box if you do not want to view the readme file.
9. Click **Finish**.

   Note: If you remotely install and configure this agent and you are monitoring a local HTTPS server using the web server plugin, restart the HTTP server.
10. Once the installation is complete, proceed to "Verify installations of Response Time monitoring agents" on page 125.

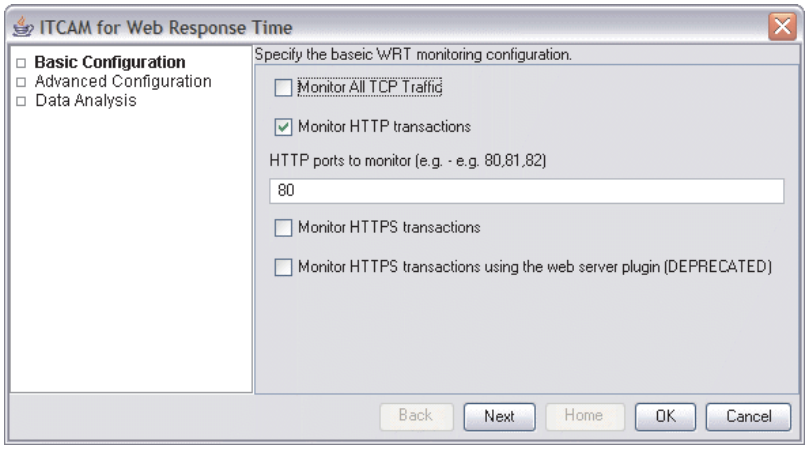
Configuring Web Response Time from the GUI (UNIX and Linux)

1. Make sure that you understand and have collected the configuration parameters for this agent from "Web Response Time parameters" on page 182 and that you have collected all the information in Information to collect before you begin installation and configuration. You use this information in completing these steps.
2. Open a command prompt and navigate to the `<install_dir>/bin` directory.

   Where `<install_dir>` is the installation location for the monitoring agent. The default installation location is `/opt/IBM/ITM`.
3. Run the **`./itmcmd manage`** command to display the Manage Tivoli Enterprise Monitoring Services window.



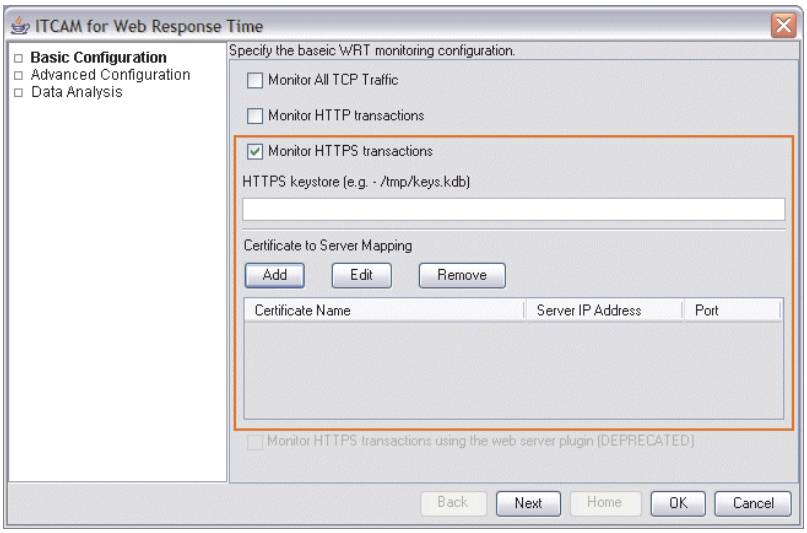
4. Right-click the ITCAM for Web Response Time agent and select **Configure** to start the basic configuration.
5. In the ITCAM for Web Response Time **Basic Configuration** window, select the appropriate checkboxes as needed and enter additional information as requested. Note that the input fields change depending on which options you select. The option to monitor all HTTP transactions is selected by default.



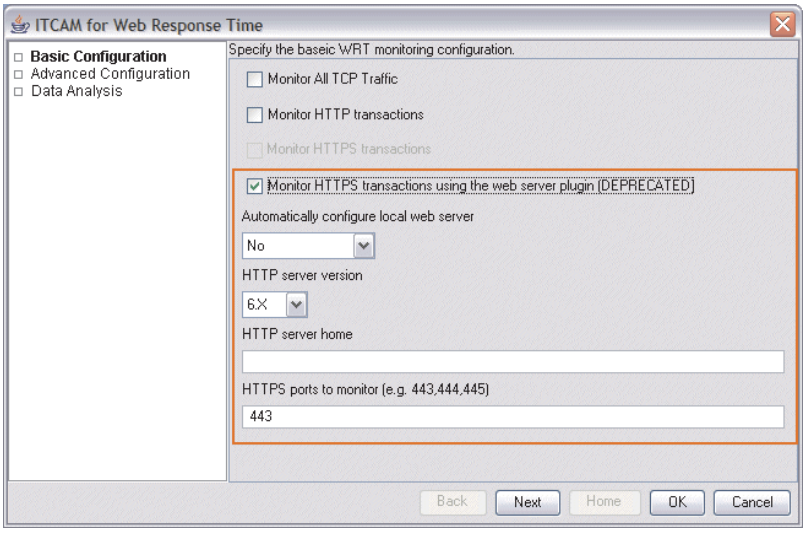
HTTP vs TCP monitoring: If you select **Monitor All TCP Traffic**, TCP traffic is monitored and response time and bandwidth related metrics for this traffic will be available in the Application, Client, and Server workspaces. Using HTTP monitoring, a web request is merged with its object requests (GIF, CSS, etc), compared to TCP monitoring, in which each of these requests is counted separately. Therefore, a single HTTP transaction can equate to many TCP transactions. So, depending on how you configure your Web Response Time agent, this information is displayed differently in the Web Response Time workspaces from the Network node in the TEP Navigator view. For more information about defining components in the Application Management Configuration Editor, see the *Administrator's Guide*.

Specify the following basic configuration information:

a. Optionally select **Monitor All TCP Traffic** to enable the agent to monitor TCP traffic.
b. Optionally select **Monitor HTTP transactions** to enable the agent to monitor HTTP traffic (this is selected by default). In the **HTTP Ports to Monitor** field, specify one or more ports to monitor, separated by commas, with no blank spaces. The default is port *80*.
c. Optionally select **Monitor HTTPS transactions** to enable the agent to monitor HTTPS traffic. Optionally, specify the following additional information:
   - Specify the HTTPS keystore containing the remote certificates of the remote HTTPS web sites to be monitored.
   - Specify the Certificate to Server Mapping, which defines which remote servers should be monitored, and the SSL certificate used by each remote server. Use the **Add**, **Edit**, and **Remove** buttons to define certificate names, server and port information.



d. Optionally select **Monitor HTTPS transactions using the web server plugin (DEPRECATED)** to enable the agent to monitor HTTPS traffic using a plugin that is installed on the local web server. Optionally, specify the following additional information:
   - Specify whether the monitoring plugin should be automatically configured on the local web server (IBM HTTP Server or Apache).
   - If you specified to automatically configure the monitoring plugin, specify the version of the selected local HTTP server.
   - Specify the home directory of the monitored HTTP server.
   - Specify one or more HTTPS ports to be monitored by the Web Response Time agent, separated by commas, with no blank spaces. The default port is *443*. See Configuring Web Response Time to monitor remote HTTPS transactions in the *Administrator's Guide* for further information.

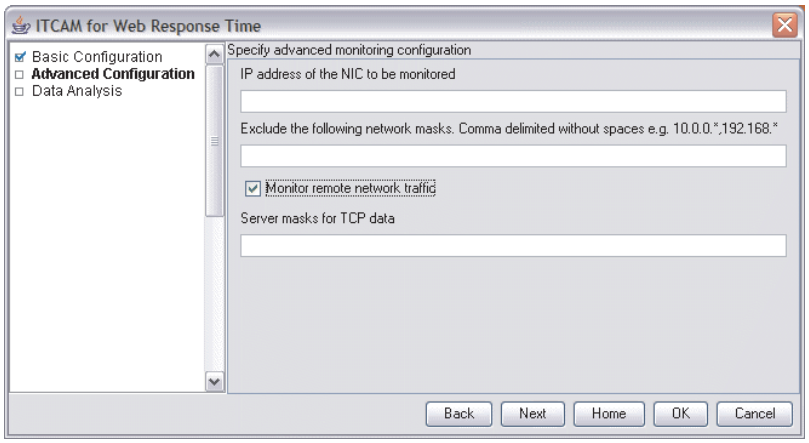


e. Click **Next** to continue to the **Advanced Configuration** window.

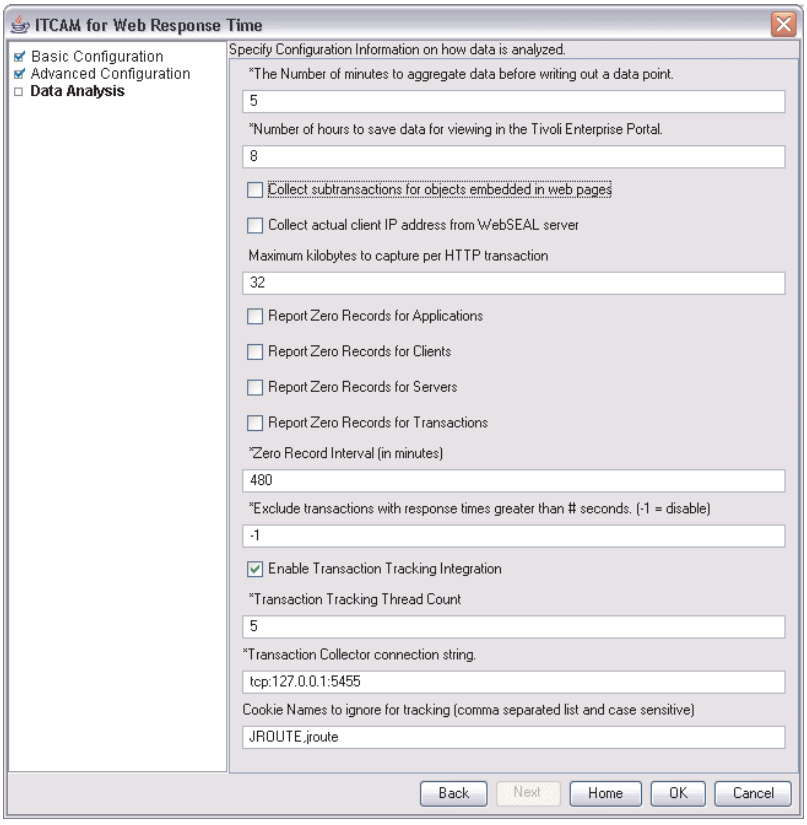
6. In the ITCAM for Web Response Time **Advanced Configuration** window, specify additional optional configuration information as needed.
   - The IP address of the selected NIC card to be monitored.

- One or more network masks to be excluded from monitoring, separated by commas, with no blank spaces. For example, you might enter *9.48.152.*,9.48.164.**, which prevents traffic on subnets 9.48.152 and 9.48.164 from being processed.
- Select the **Monitor remote network traffic** check box if you want to monitor all network traffic on the NIC. If you do not select this option, only the local traffic to and from the local IP address is monitored.
- One or more server masks for TCP data monitoring, separated by commas, with no blank spaces (for example, *10.0.0.*,192.168.**). TCP data coming from an IP address that matches one of these masks is marked as coming from a server, and not a client group.

  Use this field when you are using multiple Web Response Time agents to provide TCP tracking data to Transaction Reporter. In this scenario, to display TCP topology correctly, this data is displayed in the resulting topology view as coming from a server instead of a client. For example, suppose Web Response Time agent *WRT1* observes traffic from clients to server 9.48.152.1, and another Web Response Time agent, *WRT2*, observes traffic from server 9.48.152.1 to server 9.48.152.2. In this case, *WRT2* should be configured with a server mask that includes 9.48.152.1.



7. Click **Next** to continue to the **Data Analysis** dialog and specify additional configuration information defining how the collected data is to be analyzed.



Accept the defaults, or optionally specify the following information:

a. Specify the number of minutes to aggregate data before it is saved and summarized into an aggregate data point. The default is every 5 minutes. For example, to see data in larger aggregate intervals (for SLA reporting without diagnosis), change the default value from 5 minutes to a larger value, such as 15, 30, or 60 minutes, etc

b. Specify the number of hours that data is saved for viewing in the TEP. The default of 8 hours causes the TEP to display aggregate data collected over the last 8 hours.

c. Optionally select the check box to indicate that subtransactions should be collected for embedded objects (GIF, etc) in web pages.

d. Optionally select the check box to indicate that actual client IP addresses should be collected from a WebSEAL server.

e. Specify the maximum number of kilobytes in each HTTP request or reply. The default is 32 kilobytes.

f. Optionally select the check boxes for reporting zero records for applications, clients, servers, or transactions for the duration of the zero record interval.

g. Specify the number of minutes to continue to report zero records. the default is 480 minutes (8 hours).

h. Optionally specify a response time threshold value in seconds, which causes any transactions with response times greater than the threshold value to be excluded from all calculations. The default value is *-1*, which disables this function and includes all transactions, regardless of their response time, in the calculations.

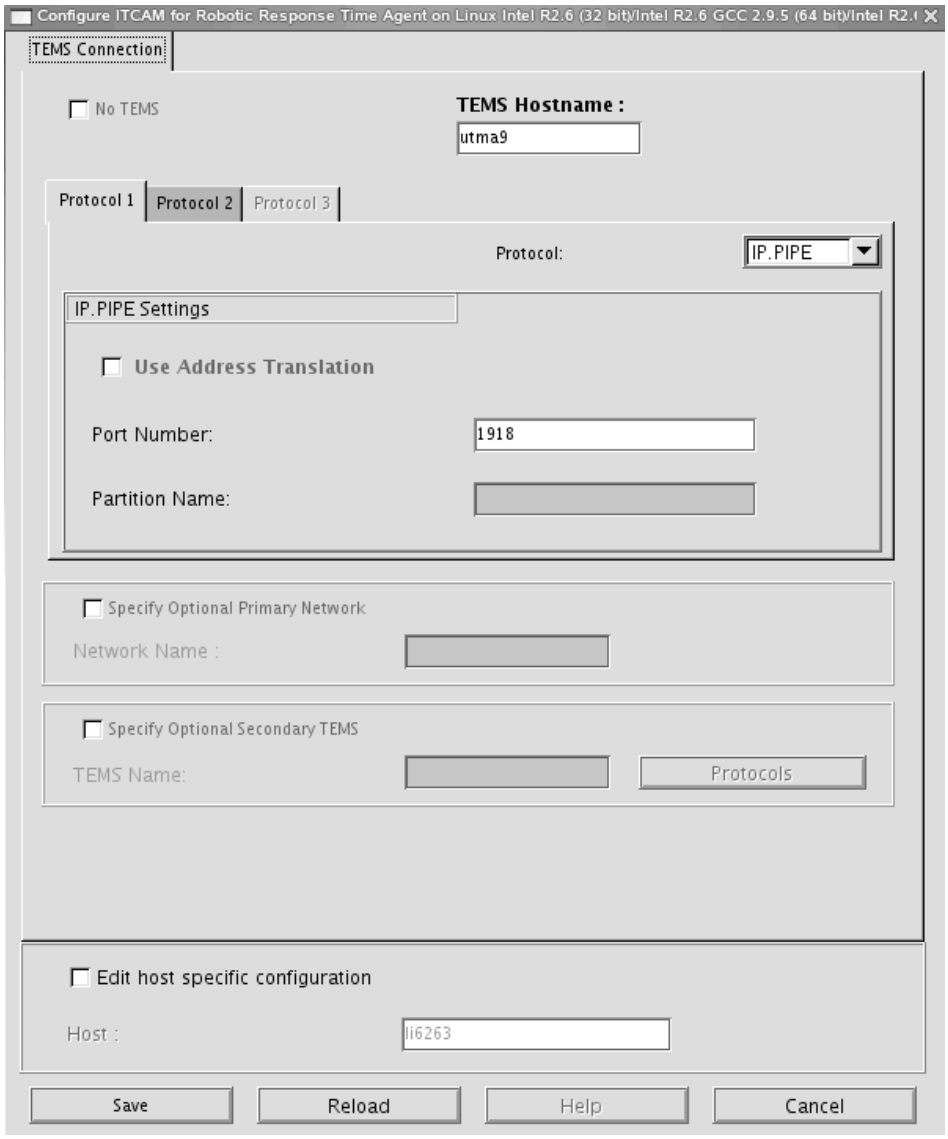
i. Optionally select the **Enable Transaction Tracking Integration** check box to cause the Web Response Time agent to send its data to the Transaction

Collector agent using generated TTAPI events for transaction tracking integration. Accept the additional default values that are displayed, or specify your own values for the following parameters:

- Transaction Tracking Thread Count (default is *5*)
- Transaction Collector connection string
- A comma separated, case-sensitive string of cookie names to ignore during tracking.

Note: You cannot enable transaction tracking integration if HTTPS (plugin mode) is enabled. The Web Response Time agent cannot obtain the necessary tracking information from transactions monitored in HTTPS plugin mode.

8. Click **OK** to display the **TEMS Connection** window.
9. Specify the TEMS connection settings.



10. Click **Save**.
11. Close the Manage Tivoli Enterprise Monitoring Services window.

12. Restart the monitoring agent. See Chapter 13, “Starting and stopping servers and agents,” on page 325.
13. Manually stop and then restart the web server.

Configuring Web Response Time on AIX systems

On AIX, the default Soft data limit is too small for the Web Response Time agent to operate effectively.

Modify the `/etc/security/limits` file to change the data segment size memory limit to unlimited (in the limits file, change the value of the `data` parameter to `-1` to set it as an unlimited value).

On any operating system, if the value of `data` is too low, the agent might shut down unexpectedly if it needs more memory than has been allocated. For more information about configuring the data segment size value, see the following Redbooks publication: http://www.redbooks.ibm.com/abstracts/TIPS0154.html.

Using the kfc_iis_install tool to configure HTTPS for IIS

Note: The HTTPS plugin method should only be used if the normal monitoring technique cannot be applied in your environment (for example, problems with encryption, keys unable to be exported, etc).

Navigate to the *ITM_Home*`\TMAITM6\wrm\CandleFilter` directory. From the command prompt, run the **`kfc_iis_install.exe`** command and follow the user prompts.

The following table outlines the usage of the **`kfc_iis_install`** tool:

Table 23. Usage of the kfc_iis_install tool

`PROGRAM (-i|-u) [-b] [-a|-f <file>] -m machine -n filtername -p filterpath`	
`-i or -u`	Install/Uninstall the filter
`-b`	A backup of the IIS settings is created before changes
`-a`	Install filter globally
`-f <file>`	Install on all websites in the file (one per line)
`-m machine`	Name of the computer, i.e. localhost
`-n filtername`	Name of the ISAPI filter
`-p filterpath`	Path of the ISAPI filter DLL

For example, you can run the tool like this:

```
>> kfc_iis_install.exe -i -b -f "weblist.txt" -m localhost -n "KFC Candle Filter" -p
"C:\IBM\ITM\tmaitm6\wrm\CandleFilter\kfcCandleFilter.dll"
```

To uninstall:

```
>> kfc_iis_install.exe -u -b -f "weblist.txt" -m localhost -n "KFC Candle Filter"
```

After installing or uninstalling, restart the W3SVC before the filter can take effect:

```
>> net stop w3svc
>> net start w3svc
```

If you do not specify `-a` or `-f <file>`, then a menu will appear so you can select the list of websites on which to install. For example:

```
Please select the website IDs (i.e., 1,3,4):
 1)  Default Web Site
 2)  Another Site
 a)  Global
 x)  Exit

>>
```

Specifying `-b` creates a backup of the IIS settings prior to any modifications. These backups are usually located at `C:\WINDOWS\system32\inetsrv\MetaBack\WRMBackup***` . To restore a previous backup, use the Internet Service Manager or the `mtaedt22.exe` tool.

Note that the kfc_iis_install.exe script is only used to install the KFC Candle filter to the relevant part of the IIS web server, or to remove it from the server. For IIS environments with HTTPS monitoring with WRM, the WRM environment files (such as kfcienv, kfcmenv, or kflmenv) and Windows registry entries must be updated manually.

Configuring Response Time from the command line (UNIX and Linux) and agent configuration parameters

This chapter describes how to configure Response Time monitoring agents from the command line in UNIX and Linux. It includes the following sections:

- The procedure for installing from the command line.
- "Application Management Console parameters" on page 178
- "Client Response Time parameters" on page 178
- "Robotic Response Time parameters" on page 179"Web Response Time parameters" on page 182

Procedure

1. Run the following command from <*install_dir*>/bin:

   ```
   ./itmcmd config -A <agent>
   ```

   where

   <*install_dir*> is the installation location for the monitoring agent.

   <*agent*> specifies the monitoring agent that you want to install:

   - **t3** specifies Application Management Console
   - **t4** specifies Client Response Time
   - **t5** specifies Web Response Time
   - **t6** specifies Robotic Response Time
2. Press **Enter**.
3. Edit the various configuration settings. See appropriate parameter section.
   - "Application Management Console parameters" on page 178
   - "Client Response Time parameters" on page 178
   - "Robotic Response Time parameters" on page 179
   - "Web Response Time parameters" on page 182
4. Press **Enter** when you are asked if the agent connects to a monitoring server.
5. Type the host name for the monitoring server.

6. Type the protocol that you want to use to communicate with the monitoring server. You have four choices: `ip`, `sna`, `ip.spipe`, or `ip.pipe`. Press **Enter** to accept the default protocol (`IP.PIPE`).
7. *(Optional)* Do one of the following:
   - To set up a backup protocol, enter that protocol and press **Enter**.
   - If you do not want to use backup protocol, press **Enter** without specifying a protocol.
8. Depending on the type of protocol you specified, provide the information you collected in Information to collect before you begin installation and configuration.
9. Press **Enter** to *not* specify the name of the `KDC_PARTITION`.
10. Press **Enter** when you are asked if you want to configure the connection to a secondary monitoring server. The default value is No.
11. Press **Enter** to accept the default for the **Optional Primary Network Name** (none).

Application Management Console parameters

Table 24. Application Management Console parameters

Parameter	Parameter for silent installation and remote deployment	What you should specify
Data Collection Configuration tab		
Data Collection Time Span (in Hours)	`KT3HRSDISP`	The maximum number of hours for reporting (default: 8)
Data Collection Frequency (in minutes)	`KT3SUMMINT`	The number of minutes for the interval (default: 5)
JMX SOAP Connector Server Configuration tab		
Port	`KT3SCPORT`	JMX SOAP connector server port (default: 1976).
SSL	`KT3SCSSL`	Whether SSL authentication is required. (default: False)
SSL Keyfile	`KT3SCSSLKEYFILE`	SSL certificate keyfile
SSL Keyfile Password	`KT3SCSSLKEYPASS`	Password to the SSL keyfile
SSL Client Authentication	`KT3SCSSLCLIENTAUTH`	Whether SSL client authentication is required (default: False)

Client Response Time parameters

Table 25. Client Response Time parameters

Parameter	Parameter for silent installation and remote deployment	What you should specify
Data Analysis Configuration tab		
Number of minutes to aggregate data before writing out a data point	`KT4OVERTIMEINTERVAL`	The time period during which the data is aggregated (default: 5)

Table 25. Client Response Time parameters (continued)

Parameter	Parameter for silent installation and remote deployment	What you should specify
Number of hours to save data for viewing in the Tivoli Enterprise Portal	KT4SUMMARYINTERVAL	The time period during which all data points are saved locally (default: 8)
Maximum number of processing threads	KT4MAXTHREADS	Maximum number of threads allowed for processing data files (default: 40)

Robotic Response Time parameters

Table 26. Robotic Response Time parameters

Parameter	Parameter for silent installation and remote deployment	What you should specify
Robotic Monitoring Configuration tab		
Playback timeout period (seconds)	TIMEOUT	A number of seconds to wait before the playback times out (default: 120)
Number of retries to attempt	NUMRET	The number of retries to attempt on timeout or availability failure (default: 1)
Lag time between retries (seconds)	RETLAG	A number of seconds to wait between retry attempts (default: 3)
Concurrent CLI playbacks	CONCUR	Whether or not CLI monitors should playback concurrently (default: True)
Abort playback on availability violation	ABRTVIOL	Whether playback should abort when an availability violation occurs (default: True)
Maximum number of concurrent robotic playbacks	KT6NUMCONCPLAYBACK	The maximum number of robotic scripts to playback concurrently (default: 10)
Script location preference	KT6SCRIPTLOCPREF	Whether to use a remotely uploaded script or a manually distributed local script when the same script exists both remotely and locally (default: Remote)
Script Depot Synchronization Interval (minutes)	KT6SCRIPTDOWNLOADINTERVAL	Specifies how often to synchronize between the local and remote script depot, checking for new and updated remote scripts to download (default: 30)

Table 26. Robotic Response Time parameters (continued)

Parameter	Parameter for silent installation and remote deployment	What you should specify
Maximum playback event buffer size	KT6MAXEVENTBUFFER	Specifies the maximum buffer size for storing playback events. (default: 1000)
Install Rational Robot Support	KT6INSTALLRATIONALROBOT SUPPORT	Specify if Rational Robot support should be installed. (default: True)
Windows Logon User Name	KT6ROBOTAUTOLOGINUSER	Specify the Windows User Name for auto re-login when rebooted (no default)
Windows Logon Password Confirm Windows Logon Password	KT6ROBOTAUTOLOGINPASSWORD	Specify the Windows Password for auto re-login when rebooted (no default)
Enable Screen and Content Capture during Playback	KT6ENABLESCREENCONTENT CAPTURE	Specify if screen and content capture should be enabled during playback. (default: False)
Rational Performance Tester Configuration tab		
Unsecure Playback Proxy Hostname	KT6PLAYBACKUNSECURE PROXYHOSTNAME	Specify the hostname of the unsecure proxy to allow the RPT scripts to playback (no default)
Unsecure Playback Proxy Port	KT6PLAYBACKUNSECUREPROXY PORT	Specify the port to use for the unsecure proxy to allow the RPT scripts to playback (default: 0)
Secure Playback Proxy Hostname	KT6PLAYBACKSECUREPROXY HOSTNAME	Specify the hostname of the secure proxy to allow the RPT scripts to playback (no default)
Secure Playback Proxy Port	KT6PLAYBACKSECUREPROXY PORT	Specify the port to use for the secure proxy to allow the RPT scripts to playback (default: 0)
Rational Robot Gui Configuration tab		
Playback per line timeout period (seconds)	KT6ROBOTPERLINETIMEOUT	A number of seconds during which a script line should complete (default: 3600)
Abort Script On Timeout	KT6ABRTONTIMEOUT	Whether robotic process should be aborted when playback times out (default: True)
Terminate Robot Process When Not Responding	KT6TERMNTWHENNOTRESP	Whether robotic process should be terminated when it is not responding (default: False)
Recovery Command When Robot Not Responding	KT6RECVRYCMDNOTRESP	The recovery command to run when Rational Robot is not responding (no default)

Table 26. Robotic Response Time parameters (continued)

Parameter	Parameter for silent installation and remote deployment	What you should specify
Reboot When Robot Not Responding	KT6REBOOTNOTRESP	Whether the computer should be rebooted when robot is not responding (default: False)
Rational Robot VU Configuration tab		
Keep VU Playback Directories	KT6KEEPVUDIR	Whether to keep the VU playback directories (default: False)
Maximum Number of Virtual Users	KT6MAXVIRTUALUSR	Maximum allowed number of virtual users. (default: 20)
Robot VU Log Level	KT6RTVUILOGLEVEL	`Rtvui` log level. (default: ERROR)
Robot VU Environment Variables	KT6VUENVVARS	`Rtvui` environment variables (no default)
Robot VU Extra CLI Parameters	KT6VUEXTRACLIPARAMS	`Rtvui` extra CLI parameters. (default: `-r`.
VU Compiler Warning Level	KT6VUCOMPWARNLEVEL	`Rtvuc` warning level (no default)
VU Compiler External Libraries	KT6VUCOMPEXTLIBS	`Rtvuc` external libraries. (default: `libSWARM32`)
VU Compiler Options	KT6VUCOMPOPTIONS	`Rtvuc` compiler options (no default)
Mercury LoadRunner Configuration tab		
LoadRunner Command Home	KT6LRCMDHOME	The home directory of the LoadRunner command (no default)
LoadRunner command	KT6LRCMD	Executable LoadRunner command. (default: `bin/mdrv.exe` on Windows.
LoadRunner command arguments	KT6LRARG0	Command arguments used to run LoadRunner (default: `-usr ${SANDBOX}${FILE_ SEPARATOR}${TRANSACTION_ RECORDING_FILE} -out ${SANDBOX} -drv_log_file ${SANDBOX} -no_drv_log`)`
Rational Functional Tester Configuration tab		
Application Configuration Preference	KT6APPCONFIGPREF	Specify whether to use Application configuration file exported with the script or the local file in RFT installed on the agent computer. (default: Exported)

Table 26. Robotic Response Time parameters (continued)

Parameter	Parameter for silent installation and remote deployment	What you should specify
Abort Script On Timeout	KT6RFTABRTONTIMEOUT	Specify whether Rational Functional Tester process should be aborted when playback times out (default: True)
Terminate RFT Process and Reboot Machine When Not Responding	KT6RFTTERMNTREBOOTWHEN NOTRESP	Specify whether the Rational Functional Tester process should be terminated and the computer rebooted, when not responding (default: False)
Data Analysis Configuration tab		
Number of minutes to aggregate data before writing out a data point	KT6OVERTIMEINTERVAL	The time period during which the data is aggregated (default: 5)
Number of hours to save data for viewing in the Tivoli Enterprise Portal	KT6SUMMARYINTERVAL	The time period during which all data points are saved locally (default: 8)
Maximum number of processing threads	KT6MAXTHREADS	Maximum number of threads allowed for processing data files. (default: 40)

Web Response Time parameters

Table 27. Web Response Time parameters

Parameter	Parameter for silent installation and remote deployment	What you should specify
Basic Configuration panel		
Monitor All TCP Traffic	KT5MONITORTCP	Whether all TCP traffic should be monitored
Monitor HTTP transactions	KT5MONITORHTTP	Whether HTTP transactions should be monitored
HTTP Ports to Monitor	KT5HTTPPORTS	If HTTP Transactions are monitored, specify HTTP ports (enter as comma separated list with no blanks, such as 80,81,82) monitored by Web Response Time (default: 80)
Monitor HTTPS transactions	KT5MONITORHTTPSAPP	Whether HTTPS transactions should be monitored
HTTPS keystore	KT5KEYSTORE	If HTTPS transactions are monitored, this keystore specifies the certificates of the remote HTTPS websites being monitored (for example, /tmp/keys.kdb)

Table 27. Web Response Time parameters (continued)

Parameter	Parameter for silent installation and remote deployment	What you should specify
Certificate to Server Mapping table	KT5SERVERMAP	If HTTPS transactions are monitored, this table maps HTTPS servers to the appropriate certificates (e.g. -cert1,server ip,server port; cert2,server2 ip;server2 port);...
Monitor HTTPS transactions using the web server plugin (DEPRECATED)	KT5MONITORHTTPS	Whether HTTPS transactions should be monitored using the web server plugin. (default: No)
Automatically configure local web server	KT5HTTPSTYPE	Whether the monitoring plugin should be automatically configured on the local web server (default: No)
HTTP Server Version	KT5IHSVER	Monitored HTTP Server version (default: 6.X).
HTTP Server Home	KT5IHSHOME	Home directory of the monitored HTTP Server
HTTPS Ports to Monitor	KT5HTTPSPORTS	HTTPS ports (comma-separated list, no blanks) monitored by Web Response Time (default: 443)
Advanced Configuration panel		
IP address of the NIC to be monitored	KT5MONITORIP	The NIC card which has the selected IP address to be monitored.
Exclude the following network masks	KT5EXCLUDEMASKS	A comma delimited list (without spaces) of network masks that should be excluded from monitoring (for example, 10.0.0.*,192.168.*)
Monitor remote network traffic	KT5REMOTE	Whether all network traffic on the NIC is monitored. (default: No)
Server masks for TCP data	KT5SERVERMASKS	TCP data coming from an IP address in these masks is marked as coming from a server, and not a client group. This is a comma delimited list (without spaces) (for example, 10.0.0.*,192.168.*).
Data Analysis panel		

Table 27. Web Response Time parameters (continued)

Parameter	Parameter for silent installation and remote deployment	What you should specify
The Number of minutes to aggregate data before writing out a data point.	KT5OVERTIMEINTERVAL	Data is aggregated for this number of minutes and then an aggregate record is saved and a new data point is created. (default: 5)
Number of hours to save data for viewing in the Tivoli Enterprise Portal.	KT5SUMMARYINTERVAL	All Data points are saved locally for this number of hours for viewing in the Tivoli Enterprise portal (default: 8)
Collect subtransactions for objects embedded in web pages	KT5COLLECTSUBTX	Specifies if subtransactions should be collected for embedded objects (default: No).
Collect actual client IP address from WebSEAL server.	KT5COLLWEBSEALCLIENTIP	Specifies if actual client IP address should be collected from a WebSEAL server (default: No)
Maximum kilobytes to capture per HTTP transaction.	KT5DATALIMIT	Specifies the maximum number of kilobytes captured in each HTTP request/reply (default: 32)
Report Zero Records for Applications	KT5ZEROAPPRECORDS	Once a Record is created continue to report on it for the duration of the Zero Record interval (default: No).
Report Zero Records for Clients	KT5ZEROCLIENTRECORDS	Once a Record is created continue to report on it for the duration of the Zero Record interval (default: No).
Report Zero Records for Servers	KT5ZEROSERVERRECORDS	Once a Record is created continue to report on it for the duration of the Zero Record interval (default: No).
Report Zero Records for Transactions	KT5ZEROTRANRECORDS	Once a Record is created continue to report on it for the duration of the Zero Record interval (default: No).
Zero Record Interval (in minutes)	KT5ZERORECORDINTERVAL	Interval (minutes) to report continue to report on zero records (default: 480)
Exclude transactions with response times greater than # seconds.	KT5EXCLUDEOUTLIERSECONDS	Exclude transactions that exceed a configurable value. By default the value for this option is -1, which includes all response times. To change it, enter a positive numeric value in seconds so that transactions exceeding this threshold value are excluded.

Table 27. Web Response Time parameters (continued)

Parameter	Parameter for silent installation and remote deployment	What you should specify
Enable Transaction Tracking Integration	KT5ENABLETRACKING	Causes the WRT Agent to send its data to the Transaction Collector Agent by generating TTAPI events for transaction tracking integration.
Transaction Tracking Thread Count	KT5TRACKINGTHREADS	The number of threads to use for Transaction Tracking Processing. (default: 5)
Transaction Collector connection string.	KT5TRANSACTIONCOLLECTOR	The connection string used to connect to a local or remote Transaction Collector.
Cookie names to ignore for tracking	KT5TRACKINGCOOKIES	Cookie names to remove from the cookie header before using for transaction tracking

Chapter 8. Installing Transaction Tracking

Transaction Tracking can be installed on UNIX, Windows, and z/OS systems.

Ensure that you read the installation considerations and prerequisites before attempting to install Transaction Tracking.

See *IBM Tivoli Composite Application Manager for Transactions Installation and Configuration Guide for z/OS* for information about installing on z/OS.

Installing Transaction Tracking on Windows systems

To install Transaction Tracking, you install the Transaction Reporter, Aggregation agents, and support files for the IBM Tivoli Monitoring components separately.

Before you begin

The separate components of Transaction Tracking can all be installed using the same InstallShield Wizard.

Note: Install all components as the same user.

The following installation procedures assume that each IBM Tivoli Monitoring component and the Transaction Reporter and Aggregation agents are installed to separate computers.

The installation procedures also assume that only the required configuration is performed during installation with configuration completed after installation.

The installation procedure for each component is the same until you select the features that you want to install on the **Select Features** window. The procedure up to this point is described here.

Before starting the installation, make sure that you have read "Installation prerequisites" on page 44.

Procedure

To launch the Transaction Tracking installation process for all components on a Windows operating system:

1. Log on as a user with administrative privileges.
2. Insert the product DVD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage/.
3. Double-click `setup.exe` to start the installation wizard.

   Tip: On Windows Server 2008 systems, if instead of the installer you see the following message, right-click the `setup.exe` file in the file explorer and select **Run as Administrator**.

   ```
   Your logon ID must have Administrator rights to install IBM Tivoli Composite
    Application Manager for Transactions
   ```
4. On the **Welcome** window, click **Next**.

5. If no IBM Tivoli Monitoring components are installed on this computer the **Prerequisites** window is displayed. Read the information and click **Next**.
6. On the **Install Prerequisites** window, options to ensure that you have the correct version of IBM GSKit or IBM Java are selected. Click **Next**. The required software is installed automatically.
7. On the **Software License Agreement** window, read the agreement and click **Accept**.
8. If you install to a computer that does not have other IBM Tivoli Monitoring components installed, the **Choose Destination Location** window with the default installation location is displayed. Change the location if required and click **Next**.
9. If you install to a computer that does not have other IBM Tivoli Monitoring components installed, the **User Data Encryption Key** window opens. Enter your own unique encryption key or accept the default and click **Next** then click **OK** in the confirmation dialog box.

   Note: You are only required to supply an encryption key if the IBM GSKit is not already installed on that computer. Use the same key across the enterprise.
10. On the **Select Features** window, select the components that you want to install. The remaining installation options depend on the features you select.

Results

See specific installation sections for more information about installing individual components.

Installing the Transaction Reporter on Windows systems

Install one or more Transaction Reporter to your IBM Tivoli Monitoring environment.

Before you begin

The Transaction Reporter is installed using the Transaction Tracking InstallShield Wizard.

Before starting the installation, make sure that you have read “Installation prerequisites” on page 44.

Procedure

To install Transaction Reporter to a Windows based system:

1. Launch the Transaction Tracking installation wizard and follow the initial procedure described in “Installing Transaction Tracking on Windows systems” on page 187.
2. On the **Select Features** window, expand **Tivoli Enterprise Monitoring Agents**, select the following features, and click **Next**:
   - **Tivoli Enterprise Monitoring Agent Framework** - installs Manage Tivoli Enterprise Monitoring Services
   - **Transaction Reporter** - installs the agent

   Note: Tivoli Enterprise Monitoring Agents may already be selected if the framework or any agents are already installed. Expand this feature and select the features you require.

3. On the **Select Program Folder** window, review the default location, update it if required, and click **Next**. This window only opens if no IBM Tivoli Monitoring components are installed on this computer.
4. On the **Agent Deployment** window, select Transaction Reporter if you want to deploy the Transaction Reporter to a remote location using the Tivoli Enterprise Monitoring Server and click **Next**.

   If you are installing locally, do not select any agents. If you are deploying agents to a remote server, select the agent to enable remote deployment.

   See Chapter 11, "Working remotely," on page 309 for further information.
5. On the **Start Copying Files** window, review the settings and if correct, click **Next**. A **Setup Status** window informs you about the progress of the installation.
6. On the **Setup Type** window, select all configuration options and click **Next**.

   You can delay some configuration until after installation if required. The following steps assume that all configuration options are selected. See "Configuring connection to the Tivoli Enterprise Monitoring Server on Windows systems" on page 217 for further information and information about configuring connections after installation.
7. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:
   a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.
   b. Select **Protocol 1** and select a protocol from the list. Several types of protocols are available: IP.UDP (uses unsecured UDP communications), IP.PIPE (uses unsecured TCP communications), IP.SPIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
   c. If additional protocols are required, select **Protocol 2** and select an second protocol from the list.
   d. Do not select **Optional Secondary TEMS Connection**. You can set up the failover support for the component later. See the *IBM Tivoli Monitoring User's Guide* for further information.
   e. Click **OK**.
8. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**.

   These settings define communications between the agent and the Tivoli Enterprise Monitoring Server. The host name or IP address of the local computer are displayed unless the Tivoli Enterprise Monitoring Server has already been specified. Ensure that you enter the host name or IP address of the Tivoli Enterprise Monitoring Server in the **Hostname or IP address** field if it is installed on another computer. The default port number for the previously selected protocol is also displayed (IP.PIPE is 1918, IS.SPIPE is 3660).
9. The **Agent Configuration** window opens. Adjust the default settings where required and click **OK**. See Transaction Reporter data collection settings in the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide* for further information.
10. The selected components are installed, configured, and started. The **InstallShield Wizard Complete** window is displayed when installation is complete. Click **Finish**.

Results

Installation of the Transaction Reporter is complete. You can now install one or more Aggregation agents such as the Transaction Collector.

Installing Transaction Collectors on Windows systems

Install one or more Transaction Collectors to your IBM Tivoli Monitoring environment, preferably to the computer on which the application that you want to monitor is installed.

Before you begin

Transaction Collectors are installed using the Transaction Tracking InstallShield Wizard.

Before starting the installation, make sure that you have read "Installation prerequisites" on page 44.

Procedure

To install a Transaction Collector on a Windows system:

1. Launch the Transaction Tracking installation wizard and follow the initial procedure described in "Installing Transaction Tracking on Windows systems" on page 187.
2. On the **Select Features** window, expand **Tivoli Enterprise Monitoring Agents**, select the following features and click **Next**:
   - **Tivoli Enterprise Monitoring Agent Framework** - installs Manage Tivoli Enterprise Monitoring Services
   - **Transaction Collector** - installs the agent

   Note: Tivoli Enterprise Monitoring Agents may already be selected if the framework or any agents are already installed. Expand this feature and select the features that you require.
3. On the **Select Program Folder** window, review the default location, update it if required, and click **Next**. This window only opens if no IBM Tivoli Monitoring components are installed on this computer.
4. On the **Agent Deployment** window, select Transaction Collector if you want to deploy the Transaction Collector to a remote location using the Tivoli Enterprise Monitoring Server and click **Next**.

   If you are installing locally, do not select any agents. If you are deploying agents to a remote server, select the agent to enable remote deployment.

   See Chapter 11, "Working remotely," on page 309 for further information.
5. On the **Start Copying Files** window, review the settings and if correct, click **Next**. Click **Yes** in the confirmation dialog box to start copying files. A **Setup Status** window informs you about the progress of the installation.
6. On the **Setup Type** window, select the required configuration options and click **Next**.

   You can delay some configuration until after installation if required. The following steps assume that all configuration options are selected. See "Configuring connection to the Tivoli Enterprise Monitoring Server on Windows systems" on page 217 for further information and information about configuring connections after installation.

7. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:
   a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.
   b. Select **Protocol 1** and select a protocol from the list. Several types of protocols are available: IP.UDP (uses unsecured UDP communications), IP.PIPE (uses unsecured TCP communications), IP.SPIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
   c. If additional protocols are required, select **Protocol 2** and select an second protocol from the list.
   d. Do not select **Optional Secondary TEMS Connection**. You can set up the failover support for the component later. See the *IBM Tivoli Monitoring User's Guide* for further information.
   e. Click **OK**.
8. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**.

   These settings define communications between the agent and the Tivoli Enterprise Monitoring Server. The host name or IP address of the local computer are displayed unless the Tivoli Enterprise Monitoring Server has already been specified. Ensure that you enter the host name or IP address of the Tivoli Enterprise Monitoring Server in the **Hostname or IP address** field if it is installed on another computer. The default port number for the previously selected protocol is also displayed (IP.PIPE is 1918, IS.SPIPE is 3660).
9. The **Agent Configuration** window opens. Adjust the default settings where required and click **OK**. See Aggregation agent data collection settings in the *Administrator's Guide* for further information.
10. The selected components are installed, configured, and started. The **InstallShield Wizard Complete** window is displayed when installation is complete. Click **Finish**.

Results

Installation of the Transaction Collector is complete. You can now install support files for the IBM Tivoli Monitoring components.

Installing support files

In addition to installing Transaction Tracking components, Aggregation agent, and Transaction Reporter, you must install support files for the IBM Tivoli Monitoring components that are used by Transaction Tracking.

Support files add support information for the predefined workspaces and situations to the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server components of IBM Tivoli Monitoring. There is a support file for each of these components.

Support files must be installed on the same computer as the components they are supporting. For example, Tivoli Enterprise Monitoring Server support should be installed on the same computer as the Tivoli Enterprise Monitoring Server.

The support files are included on the product CD or downloaded from the IBM Passport Advantage® website http://www.ibm.com/software/howtobuy/passportadvantage/ as part of the product. You install the files on Windows using

the Transaction Tracking InstallShield Wizard.

Installing Tivoli Enterprise Monitoring Server support on Windows systems

Installing Tivoli Enterprise Monitoring Server support on Windows adds application support for Transaction Tracking. Install Tivoli Enterprise Monitoring Server support to the computer on which the Tivoli Enterprise Monitoring Server (TEMS) is installed.

Procedure

To install Tivoli Enterprise Monitoring Server support:

1. Launch the Transaction Tracking installation wizard and follow the initial procedure described in "Installing Transaction Tracking on Windows systems" on page 187.
2. On the **Select Features** window, expand **Tivoli Enterprise Monitoring Server**, select the following features, and click **Next**:
   - **Transaction Collector Support**
   - **Transaction Reporter Support**
3. On the **Agent Deployment** window, select the agents that you want to remotely deploy, and click **Next**.

   If you are installing locally, do not select any agents. If you are deploying agents to a remote server, select the agent to enable remote deployment.

   See Chapter 11, "Working remotely," on page 309 for further information.
4. On the **Start Copying Files** window, review the settings and if correct, click **Next**. The files are then copied. A **Setup Status** window informs you about the progress of the installation.
5. On the **Setup Type** window, select all configuration options and click **Next**.

   You can delay some configuration until after installation if required. The following steps assume that all configuration options are selected.
6. The **Tivoli Enterprise Monitoring Server Configuration** window displays the **TEMS Type**, **TEMS Name**, and **Protocol 1** which are automatically detected. If required, specify a backup Tivoli Enterprise Monitoring Server, select **Configure Hot Standby TEMS** and select a protocol. Click **OK**. One of two windows is displayed.
7. If you have a hub Tivoli Enterprise Monitoring Server, the **Hub TEMS Configuration** window is displayed.

   The hub settings are automatically detected and depend on the protocol selected in the previous step. For all protocols except SNA, the host name or IP address and the port number of the hub Tivoli Enterprise Monitoring Server are displayed.
8. If you have a remote Tivoli Enterprise Monitoring Server, the **Remote TEMS Configuration** window is displayed. Type the name of the remote Tivoli Enterprise Monitoring Server in the **Hostname or IP Address** field.
9. In either window, click **OK**.
10. On the **Add application support to the TEMS** window, select **On this computer** as the location to which the support files should be added and click **OK**.
11. On the **Select application support to add to the TEMS** window, select the following and click **OK**:
    - **Transaction Reporter Support**

- **Transaction Collector Support**

12. On the **Application support addition complete** window, review the installation summary and click **Next**.
13. On the **Configuration Defaults for Connecting to a TEMS** window, specify the Tivoli Enterprise Monitoring Server connection information and click **OK**:
    a. Select **Connection must pass through firewall** if the agent and the Tivoli Enterprise Monitoring Server are on different sides of a firewall.
    b. Select **Protocol 1** and select a protocol from the list. Several types of protocols are available: IP.UDP (uses unsecured UDP communications), IP.PIPE (uses unsecured TCP communications), IP.SPIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
    c. If additional protocols are required, select **Protocol 2** and select an second protocol from the list.
    d. Do not select **Optional Secondary TEMS Connection**. You can set up the failover support for the component later. See the *IBM Tivoli Monitoring User's Guide* for further information.
    e. Click **OK**.
14. In the summary Configuration Defaults for Connecting to a TEMS window, check the information and click **OK**.

    These settings define communications between the agent and the Tivoli Enterprise Monitoring Server. The host name or IP address of the local computer are displayed unless the Tivoli Enterprise Monitoring Server has already been specified. Ensure that you enter the host name or IP address of the Tivoli Enterprise Monitoring Server in the **Hostname or IP address** field if it is installed on another computer. The default port number for the previously selected protocol is also displayed (IP.PIPE is 1918, IS.SPIPE is 3660).
15. The installer stops the IBM Tivoli Monitoring services before installing, configuring, and starting the agents, and restarting the services. Click **Finish**.
16. Reboot the Tivoli Enterprise Monitoring Server.

Results

Installation of Tivoli Enterprise Monitoring Server support is complete.

Installing Tivoli Enterprise Portal Server support on Windows systems

Install Tivoli Enterprise Portal Server support to the computer on which the Tivoli Enterprise Portal Server is installed.

Procedure

To install Tivoli Enterprise Portal Server support on Windows:

1. Launch the Transaction Tracking installation wizard and follow the initial procedure described in "Installing Transaction Tracking on Windows systems" on page 187.
2. On the **Select Features** window, expand Tivoli Enterprise Portal Server, select the features listed, and click **Next**:
   - **Transaction Collector Support**
   - **Transaction Reporter Support**
3. On the **Agent Deployment** window, deselect the agents, and click **Next**.

If you are installing locally, do not select any agents. If you are deploying agents to a remote server, select the agent to enable remote deployment.

See Chapter 11, "Working remotely," on page 309 for further information.

4. On the **Start Copying Files** window, review the settings and if correct, click **Next**, then click **Yes** in the Confirmation dialog box to continue the installation. The files are then copied. A **Setup Status** window informs you about the installation's progress.
5. On the **Setup Type** window, select the required configuration options and click **Next**.

   The following steps assume that you select only **Configure Tivoli Enterprise Portal**. You can delay configuration of the agents until after installation if required. See "Configuring connection to the Tivoli Enterprise Monitoring Server on Windows systems" on page 217 for further information.
6. On the **TEPS Hostname** window, enter the host name of the Tivoli Enterprise Portal Server and click **Next**. The installation continues and a **Setup Status** window informs you of its progress.
7. The installer stops the IBM Tivoli Monitoring services, before installing, configuring, and starting the agents, and restarting the services. Click **Finish**.
8. Reboot the Tivoli Enterprise Portal Server.

Results

Installation of Tivoli Enterprise Portal Server support is complete. You must now reconfigure the Tivoli Enterprise Portal Server.

Reconfiguring Tivoli Enterprise Portal Server on Windows systems:

After you have installed Tivoli Enterprise Portal Server Support on Windows you must reconfigure the Tivoli Enterprise Portal Server.

Before you begin

After you have completed the installation of Tivoli Enterprise Portal Server support, reconfigure the Tivoli Enterprise Portal Server using **Manage Tivoli Enterprise Monitoring Services**.

Procedure

To reconfigure the Tivoli Enterprise Portal Server using **Manage Tivoli Enterprise Monitoring Services**:

1. Start **Manage Tivoli Enterprise Monitoring Services** on the computer which has the Tivoli Enterprise Portal Server installed.
2. Right click **Tivoli Enterprise Portal Server** and select **Configure**.
3. In the **Configure** dialog box, select **Save** (no changes are required).

Installing Transaction Tracking support on the Tivoli Enterprise Portal desktop client on Windows systems

Install Tivoli Enterprise Portal support to the computer on which the Tivoli Enterprise Portal (TEP) desktop client is installed.

Procedure

To install Transaction Tracking support on the Tivoli Enterprise Portal desktop client on Windows:

1. Launch the Transaction Tracking installation wizard and follow the initial procedure described in "Installing Transaction Tracking on Windows systems" on page 187.
2. On the **Select Features** window, expand **Tivoli Enterprise Portal Desktop Client**, select the features listed below, and click **Next**:
   - **Transaction Collector Support**
   - **Transaction Reporter Support**
3. On the **Agent Deployment** window, deselect the agents, and click **Next**.

   If you are installing locally, do not select any agents. If you are deploying agents to a remote server, select the agent to enable remote deployment.See Chapter 11, "Working remotely," on page 309 for further information.
4. On the **Start Copying Files** window, review the settings and if correct, click **Next** and click **Yes** in the Confirmation dialog box. The files are then copied. A **Setup Status** window informs you about its progress.
5. On the **Setup Type** window, select the required configuration options and click **Next**.

   The following steps assume that you select only **Configure Tivoli Enterprise Portal**. You can delay configuration of the agents until after installation if required. See "Configuring connection to the Tivoli Enterprise Monitoring Server on Windows systems" on page 217 for further information.
6. On the **TEPS Hostname** window, enter the host name of the Tivoli Enterprise Portal Server and click **Next**. The installation continues and a **Setup Status** window informs you of the progress.

   Note: The Tivoli Enterprise Portal Server host name is automatically detected if the Tivoli Enterprise Portal Server is installed on the same computer as the Tivoli Enterprise Portal.
7. The installer stops the IBM Tivoli Monitoring services, before installing, configuring, and starting the agents, and restarting the services. Click **Finish**.
8. Reboot the Tivoli Enterprise Portal Server.

Results

Installation of Transaction Tracking support for the Tivoli Enterprise Portal desktop client is complete. If all other support files are installed, and the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server are configured, you are ready to start using Transaction Tracking.

Silent installation on Windows systems

Use the silent installation method to define the options for installing Transaction Tracking in an installation response file, and then run the installation process from the command line without interactive input. This method is useful for performing repeated installations.

About this task

A sample installation response file, `silent.txt`, is available with the product installer. The file contains comprehensive instructions on how to modify and use it.

Procedure

To run a silent installation:

1. Open the installation response file in a text editor.
2. Uncomment and modify the installation options as required, and then save the file.
3. From a command prompt, change directory to the location of the installer, `setup.exe`, and execute the command:

   ```
   start /wait setup /z"/sfresponse-file" /s /f2"log-file"
   ```

   where *response-file* is the absolute path of the installation response file, and *log-file* is the absolute path to a log file where the installer writes the results of the installation process.

Installing on UNIX systems

To install Transaction Tracking, you install the Transaction Reporters, Aggregation agents, and support files for the IBM Tivoli Monitoring components separately and then configure the components.

Before you begin

The installation procedure for each component is the same until you select the components that you want to install. The procedure up to that point is described here. Depending on your chosen deployment, you may be able to combine some of the installation tasks.

Note: Install all components as the same user.

Procedure

To install any Transaction Tracking component on a UNIX system separate from IBM Tivoli Monitoring:

1. Log in as the same user used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the `./install.sh` command and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default (`/opt/IBM/ITM`) or type the full path to a different directory.

6. If the installation directory does not already exist, you are asked if you want to create it. Type 1 to create this directory and press **Enter**.

   If any IBM Tivoli Monitoring components are currently running, the installer stops them, and restarts them when installation is complete. To stop any running components, type 1 and press **Enter**. Installation cannot continue if you choose not to stop any running components.
7. Type 1 when prompted to `Install products to the local host` and press **Enter**.
8. The software license agreement is displayed. Read the license agreement, type 1 to accept the agreement and press **Enter**.
9. If you are prompted to install prerequisite product packages, type 1 and press **Enter**.
10. If IBM GSKit is not already installed on the computer you are prompted to provide an encryption key.

    Use the same key across the enterprise. Either type the key or accept the default and press **Enter**.
11. A list is displayed of available operating systems. Type the number for the operating system that you are installing on. The default value is your current operating system. Press **Enter**.
12. Type 1 to confirm the operating system and press **Enter**. A numbered list of products available for installation is displayed.
13. Type the number corresponding to the components that you want to install. The remaining installation options depend on the features you select.

Results

See specific installation sections for more information about installing individual components and the support files.

Installing the Transaction Reporter on UNIX systems

Install one or more Transaction Reporter to your IBM Tivoli Monitoring environment.

Before you begin

The Transaction Reporter requires the Tivoli Enterprise Services User Interface component. If you are installing the Transaction Reporter on a computer that is already running other IBM Tivoli Monitoring agents, the Tivoli Enterprise Services User Interface is already installed.

Before starting the installation, make sure that you have read “Installation prerequisites” on page 44.

Procedure

To install the Transaction Reporter on a UNIX system:

1. On the host to which you want to install Transaction Reporter, run the Transaction Tracking installer and follow the initial procedure described in “Installing on UNIX systems” on page 196.
2. When you are able to select features to install, type the numbers corresponding to the following features and press **Enter**:
   - For **Transaction Reporter**, type 5.

- For **Tivoli Enterprise Services User Interface**, type 3.

3. Type 1 to confirm your selection.
4. At the prompt `Do you want to install additional products or product support packages`, type 2 and press **Enter**.

Results

You can now install the Aggregation agent such as the Transaction Collector, the support files, and then configure the product components.

Installing Transaction Collectors on UNIX systems

Install one or more Transaction Collectors to your IBM Tivoli Monitoring environment.

Before you begin

The Transaction Collector requires the Tivoli Enterprise Management Agent Framework, Tivoli Enterprise Services User Interface, and ITCAM File Transfer Enablement components. If you are installing the Transaction Collector on a computer that is already running other IBM Tivoli Monitoring agents, the Tivoli Enterprise Services User Interface component is already installed.

Before starting the installation, make sure that you have read "Installation prerequisites" on page 44.

Procedure

To install a Transaction Collector on a UNIX system:

1. On the host to which you want to install Transaction Collector, run the Transaction Tracking installer and follow the initial procedure described in "Installing on UNIX systems" on page 196.
2. When you are able to select features to install, type the numbers corresponding to the following features and press **Enter**:
   - **Transaction Collector**
   - **ITCAM File Transfer Enablement**
   - **Tivoli Enterprise Services User Interface**
3. Type 1 to confirm your selection.
4. At the prompt `Do you want to install additional products or product support packages`, type 2 and press **Enter**.

Results

When you have installed both the Transaction Reporter and Transaction Collector, install the support files, then configure the product components.

Installing support files

In addition to installing Transaction Tracking components, Aggregation agent, and Transaction Reporter, you must install support files for the IBM Tivoli Monitoring components that are used by Transaction Tracking.

Support files add support information for the predefined workspaces and situations to the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server components of IBM Tivoli Monitoring. There is a support file for each of these components.

Support files must be installed on the same computer as the components they are supporting. For example, Tivoli Enterprise Monitoring Server support should be installed on the same computer as the Tivoli Enterprise Monitoring Server.

The support files are included on the product CD or downloaded from the IBM Passport Advantage® website http://www.ibm.com/software/howtobuy/passportadvantage/ as part of the product. You install the files on Windows using the Transaction Tracking InstallShield Wizard.

Installing Tivoli Enterprise Monitoring Server support on UNIX systems

Install Tivoli Enterprise Monitoring Server support to the computer on which the Tivoli Enterprise Monitoring Server (TEMS) is installed.

Procedure

To install Tivoli Enterprise Monitoring Server support on UNIX systems:

1. Log in as the same user used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the command `./install.sh` and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default (`/opt/IBM/ITM`) or type the full path to a different directory.
6. Type `1` when prompted to `Install products to the local host` and press **Enter**.
7. Press **Enter** to display the software license agreement.
8. Type `1` to accept the license agreement and press **Enter**. A list is then displayed of available operating systems and component support categories.
9. Type the number for Tivoli Enterprise Monitoring Server (TEMS) Support and press **Enter**.
10. Type `1` to confirm your selection and press **Enter**. A list is displayed of products for which support files are to be added.
11. Type `3` to install all Transaction Tracking support files and press **Enter**.
12. Type `1` to confirm your selection and press **Enter**.
13. At the prompt `Do you want to install additional products or product support packages`, type `2` to decline the installation of more packages or support files and press **Enter**.

    Note: If Tivoli Enterprise Portal and Tivoli Enterprise Portal Server are installed on the same computer as the Tivoli Enterprise Monitoring Server,

you can type `y` at the prompt `Do you want to install additional products or product support packages` and press **Enter**, then follow the prompts to install the Tivoli Enterprise Portal and Tivoli Enterprise Portal Server support files. In a distributed IBM Tivoli Monitoring environment, install the support files separately for each component.

14. At the prompt `Do you want to seed product support on the Tivoli Enterprise Monitoring Server`, type `1` to add support to the Tivoli Enterprise Monitoring Server and press **Enter**.

    Note: The Tivoli Enterprise Monitoring Server restarts after Transaction Tracking support is added.

Results

Installation of Tivoli Enterprise Monitoring Server support is complete. You can now add Tivoli Enterprise Monitoring Server application support. See "Adding Tivoli Enterprise Monitoring Server application support on UNIX systems" for information about how to do this.

What to do next

Configure any locally installed IBM Tivoli Monitoring products using the `/opt/IBM/ITM/bin/itmcmd config` command.

Adding Tivoli Enterprise Monitoring Server application support on UNIX systems:

After you have installed Tivoli Enterprise Monitoring Server support to UNIX systems, you must add application support.

About this task

You can add application support from the command line or from **Manage Tivoli Enterprise Monitoring Services**.

To add application support for the Transaction Reporter from the command line, run the command: `/opt/IBM/ITM/bin/itmcmd support -t` *`temsname`* `to`.

To add application support for the Transaction Collector from the command line, run the command: `/opt/IBM/ITM/bin/itmcmd support -t` *`temsname`* `tu`.

Procedure

To add application support using the **Manage Tivoli Enterprise Monitoring Services** window:

1. In **Manage Tivoli Enterprise Monitoring Services**, right click Tivoli Enterprise Monitoring Server.
2. Select **Install Product Support** > **Advanced**, then in the **Install Product Support** dialog select **Transaction Collector** and **Transaction Reporter**, and click **Install**.

Installing Tivoli Enterprise Portal Server support on UNIX systems

Install Tivoli Enterprise Portal Server support to the computer on which Tivoli Enterprise Portal Server (TEPS) is installed.

Procedure

To install Tivoli Enterprise Portal Server support on UNIX systems:

1. Log in as the same user used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the command `./install.sh` and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default (`/opt/IBM/ITM`) or type the full path to a different directory.
6. If the installation directory does not already exist, you are asked if you want to create it. Type `y` to create this directory and press **Enter**.
7. At the prompt `Install products to the local host`, type `1` and press **Enter**.
8. Press **Enter** to display the agreement.
9. Type 1 to accept the agreement and press **Enter**. A list is then displayed of available operating systems and component support categories.
10. Type the number for Tivoli Enterprise Portal Server support and press **Enter**.
11. Type 1 to confirm your selection and press **Enter**. A list is displayed of products for which support files are to be added.
12. Type 3 to install all Transaction Tracking support files and press **Enter**.
13. Type 1 to confirm your selection and press **Enter**.
14. At the prompt `Do you want to install additional products or product support packages`, type 1 and press **Enter**.
15. If the system is running Tivoli Enterprise Portal Server, you can elect to install Tivoli Enterprise Portal Server Browser Client support. From the list of component support categories, type the number for Tivoli Enterprise Portal Browser Client support and press **Enter**.
16. Repeat steps 12 through step 14.
17. At the prompt `Do you want to install additional products or product support packages`, type `2` and press **Enter**.

Results

Installation of Tivoli Enterprise Portal Server support is complete. You can now configure the Tivoli Enterprise Portal Server. See “Configuring Tivoli Enterprise Portal Server on UNIX systems” on page 202 for further information.

Configuring Tivoli Enterprise Portal Server on UNIX systems:

After you have installed Tivoli Enterprise Portal Server support on UNIX systems you must reconfigure the Tivoli Enterprise Portal Server.

Before you begin

After you have completed the installation of Tivoli Enterprise Portal Server support using the command line, reconfigure the Tivoli Enterprise Portal Server using the command line or **Manage Tivoli Enterprise Monitoring Services**.

To reconfigure Tivoli Enterprise Portal Server from the command line:

1. Run the following command on the computer where the Tivoli Enterprise Portal Server is installed:

   ```
   /opt/IBM/ITM/bin/itmcmd config -A cq
   ```

2. (Linux only) Run the following command on the computer where the Tivoli Enterprise Portal Desktop Client is installed:

   ```
   /opt/IBM/ITM/bin/itmcmd config -A cj
   ```

Procedure

To reconfigure the Tivoli Enterprise Portal Server using **Manage Tivoli Enterprise Monitoring Services**:

1. Start **Manage Tivoli Enterprise Monitoring Services** on the computer where the Tivoli Enterprise Portal Server is installed.
2. Right click **Tivoli Enterprise Portal Server** and select **Configure**.
3. In the **Configure** dialog box, select **Save** (no changes are required).

Installing Transaction Tracking support for Tivoli Enterprise Portal Desktop client on Linux systems

Install Transaction Tracking support for Tivoli Enterprise Portal Desktop client to the system on which the Tivoli Enterprise Portal (TEP) desktop client is installed.

Procedure

To install Transaction Tracking support for Tivoli Enterprise Portal Desktop client on Linux:

1. Log in as the same user used for the installation of IBM Tivoli Monitoring.
2. Insert the product CD or download the product from the IBM Passport Advantage® website: http://www.ibm.com/software/howtobuy/passportadvantage.
3. Mount the installation image.
4. In the mount directory, run the command `./install.sh` and press **Enter**.
5. When prompted for the IBM Tivoli Monitoring home directory, press **Enter** to accept the default (`/opt/IBM/ITM`) or type the full path to a different directory.
6. If the installation directory does not already exist you are asked if you want to create it. Type `y` to create this directory and press **Enter**.
7. At the prompt `Install products to the local host`, type `1` and press **Enter**.
8. Type the number that corresponds to the language in which you want to display the software license agreement and press **Enter**.
9. Press **Enter** to display the agreement.

10. Type 1 to accept the agreement and press **Enter**. A list is then displayed of available operating systems and component support categories.
11. Type the number corresponding to Tivoli Enterprise Portal Desktop Client support and press **Enter**.
12. Type 1 to confirm your selection and press **Enter**. A list is displayed of products for which support files are to be added.
13. Type 3 to install all Transaction Tracking support files and press **Enter**.
14. Type 1 to confirm your selection and press **Enter**.
15. At the prompt `Do you want to install additional products or product support packages`, type 2 and press **Enter**.

Results

Installation of Transaction Tracking support for Tivoli Enterprise Portal Desktop client is complete. On Linux systems you must now reconfigure the Tivoli Enterprise Portal.

Configuring Tivoli Enterprise Portal on Linux systems:

After you have installed Tivoli Enterprise Portal Desktop Support on a Linux system you must reconfigure the Tivoli Enterprise Portal.

Before you begin

Reconfigure the Tivoli Enterprise Portal Desktop Client after you have completed the installation of Tivoli Enterprise Portal Desktop support using the command line or **Manage Tivoli Enterprise Monitoring Services**.

To reconfigure Tivoli Enterprise Portal Desktop Client from the command line, run the following command on the computer where the Tivoli Enterprise Portal Desktop Client is installed:

```
/opt/IBM/ITM/bin/itmcmd config -A cj
```

Procedure

To reconfigure the Tivoli Enterprise Portal Desktop Client using **Manage Tivoli Enterprise Monitoring Services**:

1. Open **Manage Tivoli Enterprise Monitoring Services** on the computer where the Tivoli Enterprise Portal Desktop Client is installed.
2. Right-click **Tivoli Enterprise Portal Desktop Client** and select **Configure**.
3. In the **Configure** dialog box, click **Save** (no changes are required).

Silent installation on UNIX systems

Use the silent installation method to define the options for installing Transaction Tracking in an installation response file, and then run the installation process from the command line without interactive input. This method is useful for performing repeated installations.

About this task

A sample installation response file, `silent_install.txt`, is available with the product installer. The file contains comprehensive instructions on how to modify and use it.

Procedure

To run a silent installation:

1. Open the installation response file in a text editor.
2. Uncomment and modify the installation options as required, then save the file.
3. Execute the command:

   ```
   ./install.sh -q -h home -p response-file
   ```

   where *home* is the IBM Tivoli Monitoring home directory, typically `/opt/IBM/ITM`, and *response-file* is the absolute pathname of the installation response file.

What to do next

After installation is complete, you can configure the product using the silent configuration process.

Silent configuration

After installing Transaction Tracking, you can configure it silently.

About this task

To perform silent configuration, you must create a configuration response file. Configuration response files are text files containing parameter-value pairs that specify the desired configuration settings.

A sample silent configuration file, `silent_config.txt`, is included as part of the Transaction Tracking installer.

The following syntax rules apply to configuration response files:

- Comment lines begin with a pound (#) character.
- Blank lines are ignored.
- Parameter-value pairs have the format: *PARAMETER=value*

  Do not use a space before the parameter name; you can use a space before or after an equal (=) character.

  Do not use dollar ($), equal (=), or pipe (|) characters in parameter values.

Procedure

To perform silent configuration:

1. Create a configuration response file containing the desired configuration settings.

   See `silent_config.txt` for an example.
2. Execute the command:

   ```
   /opt/IBM/ITM/bin/itmcmd config -A -p response-file prod-code
   ```

   where *response-file* is the absolute path and filename of the configuration response file, and *prod-code* is the product identifier of the component, for example `to` for Transaction Reporter or `tu` for Transaction Collector.

Upgrading from Response Time Tracking 6.1

You can upgrade to Transaction Tracking from Response Time Tracking 6.1 by using the RTT migration tool.

The tool is used to migrate ARM filter configuration data from an Response Time Tracking 6.1 environment to Transaction Tracking. It converts the Response Time Tracking filter configuration data into an Application Management Configuration profile. Migrate the data before or after you have installed Transaction Tracking.

Note: Before migrating the data, ensure that Application Management Configuration (t3 agent) is installed.

To migrate filter configuration information from Response Time Tracking 6.1:

1. Export policies from the Response Time Tracking 6.1 Management Server:
   a. Ensure that JAVA_HOME is set to your Java installation. If Java was installed by IBM Tivoli Monitoring, the default locations are:
      - For Windows systems, `C:\Program Files\IBM\Java50\jre\bin`
      - For UNIX systems, `/opt/IBM/ITM/JRE/platform/bin` where *platform* is the code for the platform on which IBM Tivoli Monitoring is installed. For example, *sol283* for Solaris version 8, 32 bit.

   b. Change directory to a temporary location such as `/tmp/rttexport`.
   c. Run the command: **/opt/IBM/itcam/RTT/MS/scripts/cli/tmptcli.sh -GetAllAgentGroups**
   d. Run the command: **/opt/IBM/itcam/RTT/MS/scripts/cli/tmptcli.sh -GetAllPolicyGroups**
   e. Run the command: **/opt/IBM/itcam/RTT/MS/scripts/cli/tmptcli.sh -GetAllManagementPolicies**

   A set of export directories are saved to the location you specified in step 1b. In this case `/tmp/rttexport/dat/*`.

   See Response Time Tracking 6.1 documentation for more detailed information.
2. Run the RTT migration tool batch file: `%CANDLE_HOME%\tmaitm6\tuspport\RTTMigrationTool.bat` *policy_file agentgroups_dir*

   where:

   - *policy_file* is the full path to the `policy`. For example, `/tmp/rttexport/dat/policies/MyListener.xml`
   - *agentgroups_dir* is the full path to the `agentgroups` directory. In this case, `/tmp/rttexport/dat/agentgroups`.

   This imports the configuration into the Application Management Configuration. You can view the policy as a profile in the Application Management Configuration Editor.
3. Repeat step 2 for each policy in the export directory `/tmp/rttexport/dat/policies/`. For example, if there are five policies in `/tmp/rttexport/dat/policies/*.xml`, run the RTT migration tool five times.
4. Log into the Tivoli Enterprise Portal and in the Application Management Configuration Editor, edit the filters for the profile if required.

   In particular, policies that had complex regular expressions that were valid in Response Time Tracking 6.1 may not be valid profiles in Transaction Tracking. Transaction Tracking supports literal characters and asterisk (*) wildcards.

See *Response Time Administrator's Guide* at http://publib.boulder.ibm.com/infocenter/tivihelp/v24r1/index.jsp?topic=/com.ibm.itcamt.doc_7.1.0.2/welcome.htm for valid expressions for filters.

Upgrading your Transaction Tracking installation

Upgrade Transaction Tracking components to the latest release using an installer. The installer updates the Transaction Reporter, Transaction Collectors, and their IBM Tivoli Monitoring support files. Run the installer on each computer that has Transaction Tracking components installed.

Before you begin

For information about obtaining the latest version, see the **Download information** on the required version of the ITCAM for Transactions Information Center on Documentation Central.

Note: Ensure that you upgrade all of the following IBM Tivoli Composite Application Manager for Transactions components to the latest release on all relevant computers:

- Transaction Collector
- Transaction Reporter
- Tivoli Enterprise Monitoring Server support
- Tivoli Enterprise Portal Server support
- Tivoli Enterprise Portal support

Also upgrade the required Data Collector plug-ins supported domains:

- To upgrade from ITCAM for Transactions V7.1.0.0, V7.1.0.1, or V7.1.0.2 to ITCAM for Transactions V7.2.0.0 and later, use the full (base) installer.
- To upgrade from V7.2.0.0, use the upgrade (patch) installer.
- To install a new version of a Data Collector plug-in, use the full (base) installer.

Note: It is not always necessary to upgrade CICS TG Transaction Tracking If CICS TG Transaction Tracking has not been updated and you attempt to upgrade it, you may receive the following message:

```
The required core product (base.v1773) is not installed. Please check
your installation directory.
Installation failed.
Rollback complete.
```

This message indicates that CICS TG Transaction Tracking was not upgraded and that the existing installation will be used.

About this task

This procedure describes how to upgrade Transaction Tracking on Windows. The procedure is similar for the other operating systems and similar to the full installation.

Procedure

To upgrade Transaction Tracking for all components installed on a particular computer:

1. Log on as a user with administrative privileges.

2. If required, download and extract the required Transaction Tracking package. See the *Download information* for the required version of ITCAM for Transactions in the ITCAM for Transactions Information Center on Documentation Central.
3. From the package or the DVD, double-click `setup.exe` to launch the installation wizard. The **Welcome** window is displayed.
4. On the **Welcome** window, click **Next**. The **Prerequisites** window opens.
5. The **Prerequisites** window displays information about the installation. It also displays the current and required GSKit and JRE versions if the installed versions are not compatible. Click **Next** to install the correct version of IBM GSKit or IBM Java.
6. On the **Software License Agreement** window, read the agreement and click **Accept**.
7. If you install to a computer that does not have other IBM Tivoli Monitoring components installed, the **Choose Destination Location** window with the default installation location opens. Change the location if required and click **Next**.
8. If you install to a computer that does not have other IBM Tivoli Monitoring components installed, the **User Data Encryption Key** window is displayed. Enter your own unique encryption key or accept the default and click **Next** then click **OK** in the summary window..

   Note: You are only required to supply an encryption key if the IBM GSKit is not already installed on that computer. Use the same key across your enterprise.
9. On the **Select Features** window, the components installed on the computer are selected. Ensure that you update all components to the latest version. Click **Next**.
10. On the **Agent Deployment** window, select a component to deploy to a remote location using the IBM Tivoli Monitoring Server if required and click **Next**.
11. On the **Start Copying Files** window, review the settings and if correct, click **Next**. Click **Yes** in the confirmation dialog box to start copying files. A **Setup Status** window informs you about the progress of the installation.
12. On the **Setup Type** window, all configuration options are selected by default. If required, deselect the configuration options to delay configuration until after upgrade is complete. Click **Next**. If you select any configuration options the installer stops all Transaction Tracking services.
13. On the **Configuration Defaults for Connecting to a TEMS** window, check the host name of the Tivoli Enterprise Monitoring Server and click **Next**.
14. On the **Configuration Defaults for Connecting to a TEMS** summary window, check the information and click **OK**.
15. If you are updating the Transaction Reporter, the **Transaction Reporter Configuration** window is displayed. Update the Transaction Reporter configuration details if required and click **OK**.
16. If you are updating a Transaction Collector, the **Transaction Collector Configuration** window is displayed. Update the Transaction Collector configuration details if required and click **OK**.
17. The selected components are installed, configured, and started. The **InstallShield Wizard Complete** window is displayed when installation is complete. Click **Finish** and read the readme file.

What to do next

After upgrading the following Tivoli Enterprise Monitoring Agents on both Windows and UNIX systems, reconfigure the agent to reestablish connection to Tivoli Enterprise Monitoring Server. Use the Manage Tivoli Enterprise Monitoring Services console or the command `itmcmd config -A agent code`. See Chapter 9, "Configuring Transaction Tracking," on page 217 for further information.

- Transaction Collector (agent code: `tu`)
- Transaction Reporter (agent code: `to`)

Distribute any new situations to the agents:

1. In the Situation editor, select the situation that you want to distribute.
2. In the **Distribution** tab, add the agents to which you want to distribute the agent to the **Assigned** list.
3. Click **OK** to distribute the situation.

On Windows systems, if you have two entries for a Transaction Tracking domain listed in **Programs and Features** (or **Add and Remove Programs**) after upgrading from any version of ITCAM for Transactions V7.1 to V7.2 or later, remove the old entry. You can remove the old entry using either the Windows GUI or the command line. Remove the old entry using the GUI as you would for any other Transaction Tracking agent. See "Uninstalling on Windows systems" on page 211 for further information.

To remove the entry using the command line:

1. Change directory to where agent for which there are duplicate entries is installed. For example, for WASTT, change directory to `%CANDLE_HOME%\TMAITM6\ktj\wastt`.
2. Run `patchman.exe` to determine the name of the old agent:

   ```
   bin\patchman.exe
   ```

3. Using the listed name for the old entry, run the following command to remove that entry:

   ```
   bin\patchman.exe -r name_of_old_entry
   ```

Update Transaction Tracking components installed on other computers to the latest version.

Ensure that you also upgrade Tivoli Enterprise Monitoring Server support, Tivoli Enterprise Portal Server support, Tivoli Enterprise Portal support.

Verifying the Transaction Tracking installation

After you have installed Transaction Tracking you can verify that the installation is correct by logging into Tivoli Enterprise Portal and using the product.

About this task

You should see Transaction Tracking workspaces in the Tivoli Enterprise Portal under **Transaction Reporter**. You should also see data within a few minutes.

Troubleshooting the installation and configuration

If you are experiencing difficulties after installing Transaction Tracking, review this section first to help you resolve any problems.

- **The Transaction Tracking workspaces are not displayed in the Navigator.**

  If the Transaction Tracking workspaces are not visible at all, check that you have installed Tivoli Enterprise Portal Server support on the computer running the Tivoli Enterprise Portal Server.
- **The Transaction Tracking workspaces are displayed in the Navigator, but have unusual names.**

  If the workspaces are visible but have names starting with kto, check that you have installed Tivoli Enterprise Portal Desktop Client support on the computer running the Tivoli Enterprise Portal.
- **The Transaction Tracking workspaces are not available.**

  If the workspaces are visible but not available, the Data Collector plug-in has successfully run in the past but conditions have changed: either the Data Collector plug-in or Transaction Reporter are not running now or the connection information to the Tivoli Enterprise Monitoring Server has changed. Check the Data Collector plug-in and Transaction Reporter services in the Manage Tivoli Enterprise Monitoring Services window.
- **The Transaction Collector and Transaction Reporter are not available in the Navigator after updating IBM Tivoli Monitoring.**

  Always restart both the Transaction Collector and Transaction Reporter after updating IBM Tivoli Monitoring.
- **When does polling start and when should I see data in the workspaces?**

  After you have installed a Data Collector plug-in you should start to see data from that node within a few minutes. If you do not see any data, check that all components, including the Data Collector plug-in are running. Also check the connection to the Tivoli Enterprise Monitoring Server and other `ktoenv` configuration settings using Manage Tivoli Enterprise Monitoring Services.

  If you installed on a distributed system, check that you installed the correct support files on each computer.
- **There is no data in the history workspaces**

  Check that you have configured the historical data collection for the data source. In addition, check that you have configured pruning and summarization of the data.
- **A node running the Data Collector plug-in is not displayed in the Navigator.**

  If the node is not visible at all, the connection between the Data Collector plug-in and the Tivoli Enterprise Monitoring Server is not configured correctly. Reconfigure the Data Collector plug-in on that node.

  This may also occur when data from one Transaction Collector overwrites that from another. Ensure that the time for all Transaction Collectors is synchronized.
- **After editing a context mask file, some column titles are missing.**

  By default in the context mask file (`contextmask_default.cfg`), the entry `Application` exists and is copied to both **BusinessApplication** and **Component** columns in workspaces. You can edit the context mask file so that distinct component and business application names are displayed. To do this you must provide both `BusinessApplication` and `Component` entries. If you only provide one entry the other column will be blank in the workspaces.
- **MQ Tracking events are not displayed in the Tivoli Enterprise Portal.**

Ensure that the path you specified using **itmcmd config -A tu** to the context mask directory is correct. The path should not contain a trailing slash. For example, on Windows the path should be `C:\IBM\ITM\TMAITM6` by default.

- **The Transaction Collector won't start on UNIX systems using `itcmd`.**

  The agent cannot be started on UNIX systems using `itmcmd` if the current working directory is not writable.

- **Topologies are not displayed.**

  If the topology viewer is not initialized properly in the Tivoli Enterprise Portal Server, topologies may not be displayed. Reconfigure the Tivoli Enterprise Portal Server.

- **Why can't I see any data from WebSphere?**

  Ensure that you have enabled ARM, see "Enabling ARM on WebSphere" on page 301.

  If Java 2 security is enabled in WebSphere, ensure that you have set Java 2 security policies. If you do not set the Java 2 security policies for ARM libraries, the libraries cannot be loaded, and you will see an error similar to the following in the logs: *SECJ0314W: Current Java 2 Security policy reported a potential violation of Java 2 Security Permission*.

- **KBB RAS1 logging is not working on a Windows 64-bit system.**

  On all Windows 64-bit systems, the Microsoft Visual Studio 2008 runtime, `msvc90.dll`, must be installed for KBB RAS1 logs to be generated. If `msvc90.dll` is not installed, download and run the Microsoft Visual C++ 2008 Redistributable Package (x64).

- **When I attempt to install a Transaction Tracking agent, my AIX platform is not listed in the OS or component support category list.**

  You can install Transaction Tracking agents to AIX 6.1 systems, even though the operating systems are not listed. In the OS or component support category list, select the number corresponding to AIX R5.3 (32 bit) or AIX R5.3 (64 bit) instead. The agents will install successfully, but the architecture code will be `aix533` instead of the expected `aix613` or `aix616`.

Uninstalling Transaction Tracking

Uninstalling Transaction Tracking is a two-step process if it is installed on the same computer that is host to Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server and Tivoli Enterprise Portal.

You must first uninstall Transaction Tracking from the computer and then remove it from the Tivoli Enterprise Portal into which it is integrated.

If Transaction Tracking is installed in a distributed IBM Tivoli Monitoring environment, an additional step is required to remove the Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server and Tivoli Enterprise Portal support files.

Removing Transaction Tracking does not affect your IBM Tivoli Monitoring environment.

Uninstalling on Windows systems

Uninstalling Transaction Tracking from a Windows system that is also host to Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server, automatically removes the associated support files. If you have a distributed installation, ensure that you uninstall the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server support files separately from the remote systems.

Procedure

To uninstall Transaction Tracking on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add or Remove Programs**; or on Windows 2008 systems, select **Start** > **Control Panel** > **Programs and Features**.
2. Navigate to **IBM Tivoli Monitoring** and click **Remove**.
3. On the **Welcome** window of the **IBM Tivoli Monitoring** installation wizard, select **Modify** to remove only the selected features, or select **Remove** to remove all features and click **Next**.
4. Click **OK** in the information dialog box.
5. On the **Add or Remove Features** window, deselect those features you want to uninstall and click **Next**.
6. Click **OK** in the confirmation dialog box.

   Transaction Tracking is removed.
7. On the **Product Remove Complete** window, click **Finish**.

What to do next

You can now clear unused Transaction Tracking information from Tivoli Enterprise Portal and uninstall Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal support files.

Uninstalling support files

Depending on your environment you may need to manually remove support files.

In an environment where both Transaction Tracking and the IBM Tivoli Monitoring components are installed on the same computer, the support files are automatically removed when you uninstall Transaction Tracking.

In a distributed environment, where Transaction Tracking and the IBM Tivoli Monitoring components are installed on separate computers, you must manually remove the support files from the remote systems.

Uninstalling Tivoli Enterprise Monitoring Server support on Windows systems:

Uninstall both Tivoli Enterprise Monitoring Server application support and the support file from the Tivoli Enterprise Monitoring Server.

Before you begin

Remove Tivoli Enterprise Monitoring Server application support first and then remove the Tivoli Enterprise Monitoring Server support component.

To remove Tivoli Enterprise Monitoring Server application support on Windows:

1. Open the **Manage Tivoli Enterprise Monitoring Services** window on the system where Tivoli Enterprise Monitoring Server is installed.
2. Right click **Tivoli Enterprise Monitoring Server**.
3. Select **Advanced** > **Remove TEMS application support**.
4. In the **Remove application support from the TEMS** dialog box, select **On this computer** and click **OK**.
5. In the **Select the application support to remove from the TEMS** dialog box, select **Transaction Tracking** and click **OK**.

Procedure

To uninstall Tivoli Enterprise Monitoring Server support on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add and Remove Programs**.
2. Navigate to **IBM Tivoli Monitoring** and click **Change/Remove**.
3. On the **Welcome** window of the installation wizard, select **Modify** and click **Next**.
4. In the information dialog box, click **OK**.
5. On the **Add or Remove Features** window, expand **Tivoli Enterprise Monitoring Server**, deselect **Transaction Tracking** and click **Next**.
6. Follow the prompts to complete the uninstall process and click **Finish**.

Uninstalling Tivoli Enterprise Portal Server support on Windows systems:

In a distributed environment, uninstall Tivoli Enterprise Portal Server support from Tivoli Enterprise Portal Server.

Procedure

To uninstall Tivoli Enterprise Portal Server support on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add and Remove Programs**.
2. Navigate to **IBM Tivoli Monitoring** and click **Change/Remove**.
3. On the **Welcome** window of the installation wizard, select **Modify** and click **Next**.
4. In the information dialog box, click **OK**.
5. On the **Add or Remove Features** window, expand **Tivoli Enterprise Portal Server**, deselect **Transaction Tracking** and click **Next**.
6. Follow the prompts to complete the uninstall process and click **Finish**.

Uninstalling Tivoli Enterprise Portal support on Windows systems:

In a distributed environment, uninstall Tivoli Enterprise Portal support by uninstalling both Tivoli Enterprise Portal Browser Client support and Tivoli Enterprise Portal Desktop Client support from Tivoli Enterprise Portal.

About this task

On Windows, Tivoli Enterprise Portal Browser Client support is bundled with Tivoli Enterprise Portal Server support. The browser client support is uninstalled when Tivoli Enterprise Portal Server support is uninstalled.

Procedure

To uninstall Tivoli Enterprise Portal Desktop Client support on Windows:

1. Select **Start** > **Settings** > **Control Panel** > **Add and Remove Programs**.
2. Navigate to **IBM Tivoli Monitoring** and click **Change/Remove**.
3. On the **Welcome** window of the installation wizard, select **Modify** and click **Next**.
4. In the information dialog box, click **OK**.
5. On the **Add or Remove Features** window, expand **Tivoli Enterprise Portal Desktop Client**, deselect **Transaction Tracking** and click **Next**.
6. Follow the prompts to complete the uninstall process and click **Finish**.

Uninstalling Transaction Tracking on UNIX systems

First uninstall Transaction Tracking from UNIX systems, before uninstalling the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server support files.

Procedure

To uninstall Transaction Tracking on UNIX systems:

1. From a command shell, type `cd /opt/IBM/ITM/bin`.
2. Type the command `./uninstall.sh`. The name of the system on which the product is installed is displayed.
3. Type the number corresponding to the Transaction Tracking component that you want to uninstall.
4. Enter `1` to confirm your selection and press **Enter**.
5. To exit the uninstaller, enter `99` and press **Enter**.

What to do next

You can now clear unused Transaction Tracking information from Tivoli Enterprise Portal and uninstall Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal support files.

Uninstalling Transaction Tracking support files on Linux and UNIX systems

After you have uninstalled Transaction Tracking from a Linux or UNIX system, uninstall the Tivoli Enterprise Portal, Tivoli Enterprise Portal Server, and Tivoli Enterprise Monitoring Server support files separately from the remote systems.

Procedure

To uninstall Transaction Collector support files on remote Linux and UNIX systems:

1. Run the command `cinfo` and type `1` to list the product support installed.
2. Run the required command:
   - `/opt/IBM/ITM/bin/uninstall.sh tu tms` to remove Transaction Collector support for Tivoli Enterprise Monitoring Server
   - `/opt/IBM/ITM/bin/uninstall.sh tu tps` to remove Transaction Collector support for Tivoli Enterprise Portal Server
   - `/opt/IBM/ITM/bin/uninstall.sh tu tpw` to remove Transaction Collector support for Tivoli Enterprise Portal Browser Client

- `/opt/IBM/ITM/bin/uninstall.sh tu tpd` to remove Transaction Collector support for Tivoli Enterprise Portal Desktop Client

What to do next

Repeat the procedure to uninstall Transaction Reporter support files on remote Linux and UNIX systems. Run the required command:

- `/opt/IBM/ITM/bin/uninstall.sh to tms` to remove Transaction Reporter support for Tivoli Enterprise Monitoring Server
- `/opt/IBM/ITM/bin/uninstall.sh to tps` to remove Transaction Reporter support for Tivoli Enterprise Portal Server
- `/opt/IBM/ITM/bin/uninstall.sh to tpw` to remove Transaction Reporter support for Tivoli Enterprise Portal Browser Client
- `/opt/IBM/ITM/bin/uninstall.sh to tpd` to remove Transaction Reporter support for Tivoli Enterprise Portal Desktop Client

Clearing Tivoli Enterprise Portal

After you have uninstalled Transaction Tracking, you might want to clear unused Transaction Tracking information from the Tivoli Enterprise Portal.

About this task

When you remove Transaction Tracking, **Unnamed** entries remain in the Navigator.

Procedure

To clear unused Transaction Tracking information from Tivoli Enterprise Portal:

1. In the Tivoli Enterprise Portal, select **Enterprise** in the navigator.
2. Right click on **Enterprise** and select **Workspace** > **Managed System Status**.
3. In the **Managed System Status** view, right click each of the following offline entries and select **Clear offline entry**:
   - ***hostname*:T3** to clear the Application Management Console
   - ***hostname*:TO** to clear the Transaction Reporter
   - ***hostname*:TU** to clear the Transaction Collector

   The entries are cleared from the Tivoli Enterprise Portal.
4. Click **Refresh** to update the Tivoli Enterprise Portal display.

Reinstalling Transaction Tracking

Before reinstalling Transaction Tracking, check that the computer on which you are installing Transaction Tracking does not have any residual files.

On Windows systems:

1. Delete all files from the following directories:
   - `tmaitm6\todata`
   - `tmaitm6\tosupport`
   - `tmaitm6\tusupport`
2. Search for and delete the following files:
   - `plugin_th_comp-th_all_1.jar`
   - `plugin_to_comp-to_all_1.jar`

- plugin_tu_comp-tu_all_1.jar

3. Search for and delete all to_dd_071*.xml files.
4. Search for and delete all tu_dd_071*.xml files.
5. Search for and delete all ttas.dat* files.

On UNIX systems:

1. Delete all files from the following directories:
   - *platform*/to/tosupport
   - *platform*/to/todata
   - *platform*/tu/tusupport
2. Search for and delete all ttas.dat* files.

To reinstall Transaction Tracking follow the procedures described in Chapter 8, "Installing Transaction Tracking," on page 187.

Chapter 9. Configuring Transaction Tracking

Most configuration to integrate Transaction Tracking with the IBM Tivoli Monitoring environment occurs when you install Transaction Tracking.

However, some additional configuration and tuning may be required.

You can configure agents in one of two ways:

- Log into each system and manually configure the Tivoli Enterprise Management Agent

  This approach is described in this section.
- Configure the Tivoli Enterprise Management Agents remotely from a central location using the Tivoli Enterprise Portal

  See Deploying non-OS agents in the IBM Tivoli Monitoring InfoCenter for further information. Refer back to this section for the required configuration values.

Configuring connection to the Tivoli Enterprise Monitoring Server on Windows systems

The Transaction Reporter and Aggregation agents must be configured to connect to the Tivoli Enterprise Monitoring Server.

Before you begin

On Windows, these parameters are typically configured during installation of the individual agents, but you can reconfigure both the parameters and the connections manually at any time.

About this task

Each configuration generates a set log files which you can use to diagnose problems. These log files are grouped in a log folder located at `\IBM\ITM\InstallITM\plugin\executionEvents\logs\` in a default installation.

The folder is named *date-time* for when the configuration occurred.

Note: Ensure that the Tivoli Enterprise Monitoring Server is running before you configure connection to it.

The procedure for reconfiguring connection to the Tivoli Enterprise Monitoring Server is similar for the Transaction Reporter and the Aggregation agents, such as Transaction Collectors. The following procedure uses the Transaction Reporter as an example. To reconfigure a Transaction Collector, select Transaction Collector in the Manage Tivoli Enterprise Monitoring Services and follow the same procedure.

Procedure

To manually configure connection to the Tivoli Enterprise Monitoring Server on Windows:

1. Select **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services** to display the Manage Tivoli Enterprise Monitoring Services.
2. Right-click **Transaction Reporter** and select **Reconfigure**. The **Agent Advanced Configuration** window is displayed showing existing Tivoli Enterprise Monitoring Server settings. This window is named **Configuration Defaults for Connecting to a TEMS** during installation.
3. Edit these settings if required:
   a. If the Transaction Reporter and the Tivoli Enterprise Monitoring Server are on different sides of the firewall, select **Connection must pass through firewall**.
   b. In the **Protocol 1** field, select a new protocol if required. Several types of protocols are available: IP.UDP (uses unsecured UDP communications), IP.PIPE (uses unsecured TCP communications), IP.SPIPE (uses SSL secure TCP communications), and SNA (uses SNA for mainframe components).
   c. In the **Protocol 2** field, select an additional protocol if required.
   d. Select **Optional Secondary TEMS Connection** and select protocols for a backup Tivoli Enterprise Monitoring Server if required.
   e. Click **OK**.
4. On the **Agent Advanced Configuration** summary window, automatically detected settings are displayed. For all protocols except SNA, a host name or IP address and port number are displayed. Check that these Tivoli Enterprise Monitoring Server settings are correct and click **OK**.

   These settings define communications between the Transaction Reporter and the Tivoli Enterprise Monitoring Server. The host name or IP address of the local computer are displayed unless the Tivoli Enterprise Monitoring Server has already been specified. Ensure that you enter the host name or IP address of the Tivoli Enterprise Monitoring Server in the **Hostname or IP address** field if it is installed on another computer. The default port number for the previously selected protocol is also displayed (IP.PIPE is 1918, IS.SPIPE is 3660).
5. In the **Transaction Reporter Configuration** window, set the required values as described in Tuning data collection in the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide* and click **OK**. The Transaction Reporter is reconfigured. When the configuration is complete you are returned to the **Manage Tivoli Enterprise Monitoring Services** window.
6. Right-click **Transactions Reporter** and select **Start** to start the service running again.

Configuring connection to the Tivoli Enterprise Monitoring Server on UNIX systems

To configure the Transaction Reporter and the Aggregation agents to work with IBM Tivoli Monitoring, you must set connection parameters so they can contact the Tivoli Enterprise Monitoring Server. This configuration must be done manually on UNIX systems for each agent after they have been installed.

About this task

You can update your configuration at any time. A log file is generated for each configuration. Use this file to diagnose any problems.

The file is located at:

```
/opt/IBM/ITM/InstallITM/plugin/executionEvents/logs/
install_plugin_comp_is_n
```

where *n* is a number increasing by one for each generation.

Perform the configuration as the same user that was used when IBM Tivoli Monitoring was installed.

To configure the connection from the command line, run the following command from the computer where the agent is installed.

```
/opt/IBM/ITM/bin/itmcmd config -A agent abbrev
```

where *agent abbrev* is the two-letter code for the required agent:

- `to` for Transaction Reporter
- `tu` for Transaction Collector
- `t5` for Web Response Time

When configuration is complete, ensure you restart the agent using the command:

```
/opt/IBM/ITM/bin/itmcmd agent start agent abbrev
```

For parameter information, see Transaction Reporter data collection settings in the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide*.

Procedure

To configure the agent using **Manage Tivoli Enterprise Monitoring Services**:

1. Change directory to `/opt/IBM/ITM/bin`.
2. Run the command: `./itmcmd manage`. The **Manage Tivoli Enterprise Monitoring Services** window is displayed. This is an X Window System application. The steps required to configure the agents using the **Manage Tivoli Enterprise Monitoring Services** are similar to those required on Windows. See "Configuring connection to the Tivoli Enterprise Monitoring Server on Windows systems" on page 217 for further information about the procedure.

Using multiple Transaction Reporters

For ITCAM for Transactions V7.2.0.2 and later, you can use multiple Transaction Reporters to collect data from a single Aggregation agent if required. This scenario should not be necessary except in very large enterprise environments.

Specifying the shared Aggregation agent

For each Transaction Reporter, include the name of the shared Aggregation agent in the Aggregation agent List. This list is a Transaction Reporter data collection setting. Reconfigure the Transaction Reporter using the Manage Tivoli Enterprise Monitoring Services. See Transaction Reporter data collection settings for further information.

Considerations when using multiple Transaction Reporters

To minimize any problems that you may have using multiple Transaction Reporters follow these guidelines:

- Start the Transaction Reporters at the same time.

Because baselines and deviations are calculated independently at each Transaction Reporter, if you start the Transaction Reporters at the same time, you reduce the likelihood of data from the same Transaction Collector being displayed in different periods in the separate Transaction Reporters. The baseline and deviation metrics are dependent on how long each Transaction Reporter has been collecting data from the Transaction Collector. This data may vary slightly between the Transaction Reporters based on the difference in how long each Transaction Reporter has been running.

- Define only one Transaction Reporter in a situation for a specific application.

Log files in Transaction Tracking

Configuration and other information is recorded in log files. Use the log files to help diagnose problems.

Transaction Tracking uses standard IBM Tivoli Monitoring KBB RAS1 logging. Log files are grouped in a log folder located at `\IBM\ITM\logs\` in a default installation. Installation and configuration log files are located at `\IBM\ITM\InstallITM\plugin\executionEvents\logs\` in a default installation.

Each log has a maximum size of 8MB. The default logging levels are:

- For the Transaction Reporter (`KTO`): `ERROR (COMP:kto_Aggregation ALL) (COMP:kto_Baseline ALL) (COMP:kto_Command ALL) (COMP:kto_TEMA ALL) (COMP:kto_Topology ALL) (COMP:kto_XML ALL)`.
- For the Transaction Collector (KTU): `ERROR (COMP:TU ERROR) (COMP:TU_MAIN ERROR,DETAIL) (COMP:TU_TEMS ERROR,DETAIL)`.
- For the ARM Data Collector, the logging levels are dependent on the application that loads the ARM library.

You can reduce the logging detail and size of the logs by setting the KBB RAS1 filters to `ERROR` only.

Note: On all Windows 64-bit systems, the Microsoft Visual Studio 2008 runtime, `msvc90.dll`, must be installed for KBB RAS1 logging to work. If `msvc90.dll` is not installed, download and run the Microsoft Visual C++ 2008 Redistributable Package (x64).

Chapter 10. Installing and configuring Transaction Tracking Data Collector plug-ins

The Data Collector plug-ins required are determined by the systems that you want to monitor in your environment.

With the exception of MQ Tracking and Data Collector for WebSphere Message Broker that can be installed by the Transaction Tracking installer, the Transaction Tracking Data Collector plug-ins each have their own installer and are installed separately. Use the information in this section to help you determine the Data Collector plug-ins that you require.

Selecting Data Collector plug-ins to install

Table 28 describes some domain tracking scenarios and the Data Collector plug-ins required.

Table 28. Domain tracking scenarios

Scenario	Data collector plug-in required
Track WebSphere Application Server.	ARM with WASTT, or ITCAM for Application Diagnostics
Track interactions between ARM-instrumented applications on WebSphere Application Server and WebSphere MQ.	WASTT
Track transactions flowing from WebSphere Application Server through the CICS TG Gateway Daemon into CICS on z/OS.	• CICS TG Transaction Tracking • ITCAM for Application Diagnostics • CICS Tracking
Track ECI, DPL, IBM MQSeries and SOAP transactions on z/OS.	CICS Tracking
Track transactions flowing from WebSphere Application Server through the IMS to WebSphere MQ on z/OS.	• IMS Tracking • ITCAM for Application Diagnostics
Track WebSphere MQ transactions on z/OS and distributed systems.	MQ Tracking
Track transactions in .NET.	.NET Data Collector
Track transactions in the Tuxedo application.	Tuxedo Tracking
Track interactions between applications that pass through WebSphere Message Broker environments.	Data Collector for WebSphere Message Broker
Track transactions between WebSphere Message Broker and WebSphere MQ.	• Data Collector for WebSphere Message Broker • MQ Tracking
Track transactions from WebSphere Message Broker to SOA	• Data Collector for WebSphere Message Broker • ITCAM for SOA
Track interactions between SOA instances.	ITCAM for SOA with the ITCAM for SOA Log File Service

Installing Transaction Tracking Data Collector plug-ins silently on Windows systems

You can install most Transaction Tracking Data Collector plug-ins silently on Windows systems using the command line interface.

Run the following command to install a Data Collector plug-in silently:

```
dc_installer.exe silent=y installdir=C:\IBM\ITM\TMAITM6\dc_dir install_tag=
install_value
```

Where:

- *dc_installer.exe* is the name of the Data Collector plug-in installer
- *dc_dir* is the directory path to where the Data Collector plug-in is installed
- *install_tag=install_value* name, value pairs are specific to each Data Collector plug-in

  For example, for .NET Data Collector the name, value pairs can be the following values:

  – `tt_svr_adr = ttserver host name/ip`, the default host name is `ttserver.ibm.com`
  – `tt_svr_port = ttserver port`, the default port is `5455`
  – `tt_svr_protocol = tcp`
  – `monitor_now = Yes|No`, the default is `No`

Example - install .NET Data Collector silently

To install .NET Data Collector silently from the command line, run the following command:

```
dotnet-ttmonitoring-win32-32-v7300.exe silent=y installdir=C:\IBM\ITM\TMAITM6\
ktj\dotNET tt_svr_adr=ttserver.ibm.com tt_svr_port=5455 tt_svr_protocol=tcp
monitor_now=Yes
```

Preparing WebSphere MQ transaction tracking

To prepare WebSphere MQ transaction tracking, install MQ Tracking on each computer running a queue manager, and then enable transaction tracking on computers running queue managers and WebSphere MQ business applications.

About this task

MQ Tracking is available in ITCAM for Transactions V7.1 and later.

Table 29 lists the intra-domain and inter-domain tracking supported by MQ Tracking.

Table 29. MQ Tracking - domain interactions supported

Domain interactions tracked	Notes
WebSphere MQ to and from WebSphere MQ	• Channels supported in all ITCAM for Transactions versions. • Clustering and z/OS shared queues are supported in ITCAM for Transactions V7.1.0.2 and later.

Table 29. MQ Tracking - domain interactions supported (continued)

Domain interactions tracked	Notes
WebSphere Application Server to and from WebSphere MQ	• Client and bindings mode is supported if using ITCAM for Application Diagnostics. • JMS is supported. • XA and SIB are not supported.
WebSphere Application Server to WebSphere MQ	• Client and bindings mode is supported if using WASTT and ITCAM for Transactions V7.1.0.2 and later. • ARM must be enabled in WebSphere Application Server.
WebSphere MQ to and from IMS	Using the WebSphere MQ Bridge.
WebSphere MQ to and from CICS	Using the WebSphere MQ Bridge.
WebSphere Message Broker to and from WebSphere MQ	All protocols are supported.

Procedure

To prepare your environment for WebSphere MQ transaction tracking:

1. Check that your version of WebSphere MQ is supported:
   a. Link to the required ITCAM for Transactions version from ITCAM for Transactions on Documentation Central
   b. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**.
   c. In the System requirements and prerequisites page, select Transaction Tracking from the Compatible software list.
2. Identify all the computers running queue managers that process interactions between business applications.
3. On each computer running a queue manager, install and configure MQ Tracking.

   Ensure that you install MQ Tracking to the local file system, not a network file system.

   Note: When updating an existing MQ Tracking installation, stop the queue manager before commencing the installation.
4. After installation is complete, enable transaction tracking by configuring API exits on all WebSphere MQ queue managers, and deploying channel message exits to all receiver and requester channels.

   Ensure that the `ttdcmqexits.cfg` file, the exit modules, and the IPC queue directory are on the local file system, not a network file system.

   Note: Except on IBM i systems, each MQ exit contains a build number in its file name. Ensure that you update each WebSphere MQ queue manager with the new MQ exit file names whenever you upgrade your IBM Tivoli Composite Application Manager for Transactions installation.

Installing and configuring MQ Tracking

The installation and configuration process for MQ Tracking is dependent on the platform to which you are installing.

For all platforms, the basic installation procedure is as follows:

1. Install MQ Tracking.
2. Configure the API Exits on the queue managers.
3. Configure any receiver or requester channels with the channel exit.
4. Restart the queue managers and channels.

If you are upgrading your MQ Tracking installation:

1. Stop the queue managers.
2. Upgrade MQ Tracking.
3. Update the API and channel exit definitions.
4. Restart the queue managers and channels.

Installing MQ Tracking on Windows, Linux, and UNIX systems

Install MQ Tracking on each computer running a queue manager.

Before you begin

For information about obtaining the latest version of Transaction Tracking, see the Download information in the required ITCAM for Transactions Information Center.

Procedure

To install MQ Tracking on Windows, Linux, and UNIX systems:

1. Launch the Transaction Tracking installer.
2. Install the following features:
   - Tivoli Enterprise Monitoring Agent Framework
   - MQ Tracking
3. Configure MQ Tracking. See “Configuring MQ Tracking” on page 226 for further information.
4. Restart all processes that interact with your queue manager to ensure correct uptake of the new exits. This includes restarting the queue managers, channels, listeners, and any processes that connect to MQ in bindings mode.

What to do next

On Windows platforms, MQ Tracking is configured during the installation process. For other platforms, configure MQ Tracking after installation. See “Configuring MQ Tracking” on page 226 for further information.

Installing MQ Tracking on IBM i systems

Install MQ Tracking on each computer running a queue manager.

Before you begin

MQ Tracking for IBM i systems is included with the AIX media.

For information about obtaining the latest version of Transaction Tracking, see the Download information in the required ITCAM for Transactions Information Center.

Procedure

To install MQ Tracking on IBM i systems:

1. On IBM i, create a library containing sufficient empty save files to hold the uploaded files. MQ Tracking requires two files to include the Transactions Base and MQ Tracking licensed programs. To create the library and save files, use the following commands:

   ```
   CRTLIB LIB(CYT72PROD)
   CRTSAVF CYT72PROD/CYT72BASE
   CRTSAVF CYT72PROD/CYT72OPT1
   ```

2. Start an FTP session to your IBM i machine and upload the required save files with the following commands:

   ```
   ftp (your_IBM_i_hostname)
   bin
   put CYT72BASE.SAVF CYT72PROD/CYT72BASE
   put CYT72OPT1.SAVF CYT72PROD/CYT72OPT1
   ```

3. On IBM i, install Transactions Base and MQ Tracking licensed programs. Enter the **RSTLICPGM** commands, specifying the install device as *SAVF and naming the save file containing the options you want to install. For example:

   ```
   /* Base */
   RSTLICPGM LICPGM(5724S79) DEV(*SAVF) SAVF(CYT72PROD/CYT72BASE) RSTOBJ(*ALL)
     LNG(2924) OPTION(*BASE) OUTPUT(*PRINT)

   /* MQ Tracking */
   RSTLICPGM LICPGM(5724S79) DEV(*SAVF) SAVF(CYT72PROD/CYT72OPT1) RSTOBJ(*ALL)
     LNG(2924) OPTION(1) OUTPUT(*PRINT)
   ```

4. (Optional) Remove the installation save files on IBM i by running the following command:

   ```
   DLTLIB LIB(CYT72PROD)
   ```

5. Configure MQ Tracking. See “Configuring MQ Tracking” on page 226 for further information.
6. Restart all processes that interact with your queue manager to ensure correct uptake of the new exits. This includes restarting the queue managers, channels, listeners, and any processes that connect to MQ in bindings mode.

What to do next

Configure MQ Tracking after installation. See “Configuring MQ Tracking” on page 226 for further information.

Configuring MQ Tracking

After you have installed MQ Tracking, you can configure it.

Table 30 describes the configuration parameters specific to MQ Tracking.

Table 30. MQ Tracking configuration parameters

Configuration parameter name	Description
MQ Data Collector General Settings	
Disable Persistent Cache	Controls the persistence of the buffered transaction data that MQ Tracking collects. Setting this property to `true` means that the buffered data is not persistent, and will not be preserved if MQ Tracking is shut down, which can lead to lost events if MQ Tracking is restarted. Set this value to `false` unless otherwise instructed.
IPC Queue Directory	Specifies the location of the directory into which MQ transaction data is buffered by API exits and read by MQ Tracking.
IPC Message Queue Size	(IBM i, Linux, and UNIX only) Specifies the maximum capacity (in kilobytes) of the System V message queue used by the **`IPCMessageQueue`** transport. MQ Tracking must be executed with appropriate privileges, such as `root` on Linux and UNIX systems, at least once for this setting to take effect.
MQ Data Collector Log Settings	
Log Level	Sets the amount of log information written to the MQ Tracking log file; set this property as desired. The log file is located in the IBM Tivoli Monitoring logs directory.
MQ Data Collector TT Settings	
Transaction Collector Location	Specifies the protocol, IP address, and port that MQ Tracking uses to send tracking events to the Transaction Collector. For IPv4, use the format *`protocol`*`:`*`IPv4 address`*`:`*`port`*, for example `tcp:127.0.0.1:5455`. For IPv6, use the format *`protocol`*`:[`*`IPv6 address`*`]:`*`port`*, for example, `tcp:[::1]:5455` is equivalent to the previous IPv4 example. The only protocol currently supported is TCP.
Connection Timeout Value	Specifies the timeout (in milliseconds) for each attempt made by MQ Tracking to connect to the Transaction Collector.
Connection Retry Interval	Specifies the interval (in milliseconds) between attempts to connect to the Transaction Collector if a connection attempt fails.
Connection Retries	Specifies the number of attempts made by MQ Tracking to connect to the Transaction Collector. A value of 0 indicates unlimited retries.
MQ Data Collector Exits Settings	
IPC Primary Transport	(IBM i, Linux and UNIX only) Specifies which IPC transport the exits will use, either the **`IPCSharedQueue`** or the **`IPCMessageQueue`**.
MQ Exits Config Directory	(Linux, UNIX, and Windows systems only) Specifies the directory in which the API exits configuration file is installed. The maximum allowed length of this path is 32 characters. Make note of this value because it is required when configuring exits.

When configuring MQ Tracking on Windows platforms during installation, you are prompted to select a Tivoli Enterprise Monitoring Server connection; accept the defaults as this connection is not required.

To configure MQ Tracking on other platforms, run the following commands when installation is complete:

- On IBM i platforms, run the command `/QIBM/ProdData/cyt/wmq/bin/agent.sh config`
- On Linux and UNIX platforms, run the command `/opt/IBM/ITM/bin/itmcmd config -A th`

To start or stop MQ Tracking, use **Manage Tivoli Monitoring Services** or run the following commands:

- On IBM i platforms, run the command `/QIBM/ProdData/cyt/wmq/bin/agent.sh start|stop`
- On Linux and UNIX platforms, run the command `/opt/IBM/ITM/bin/itmcmd agent start|stop th`

IPCMessageQueue system configuration

For Transaction Tracking V7.2.0.1 and later, System V message queues (**IPCMessageQueue**) can be used on IBM i, Linux, and UNIX systems as the primary transport instead of **IPCSharedQueue**.

The primary transport is the main mechanism used by the API exits to convey MQ transaction data to the MQ Tracking agent.

If System V message queues are used, **IPCSharedQueue** is instead used as a backup transport for key messages when the System V message queue is not available.

For Linux and UNIX, only use the **IPCMessageQueue** transport if both:

- System V kernel settings are satisfied
- Sufficient System V message queue resources are available in addition to existing usage.

  To check the current System V message queue usage on a system, use the command **ipcs -q**.

System V kernel settings for Linux and UNIX systems

Use the System V kernel settings described in Table 31 as a guide for your operating system.

Table 31. System V settings for Linux and UNIX systems

System V setting	Description	Setting	Notes
`msgmni` (Applies to HP-UX 11i, Solaris 9, Solaris 10, and Linux)	Limits the number of message queues in the system.	The MQ Tracking data collector uses only one System V message queue.	No changes to the default are required.
`msgmnb` (Applies to all supported Linux and UNIX systems)	The maximum queue size for new queues.	No change required to `msgmnb`, but the maximum size of the MQ Tracking System V queue must be increased.	Run the MQ Tracking agent as `root` at least once to override the default with **IPCMessageQueueSize**.
`msgmax` (Applies to HP-UX 11i, Solaris 9, Solaris 10, and Linux)	Limits the maximum size of a message.	Set to `2048` KB or larger.	No changes to the default are required.

Table 31. System V settings for Linux and UNIX systems (continued)

System V setting	Description	Setting	Notes
`msgseg` (Applies to HP-UX 11i and Solaris 9)	Controls the number of message blocks used by the kernel to store messages.	Set to `32767` or larger.	HP-UX 11i and Solaris 9 defaults are insufficient.
`msgssz` (Applies to HP-UX 11i and Solaris 9)	Controls the size of memory blocks used by the kernel to store values. Messages do not share blocks.	Set to `1024` or larger.	HP-UX 11i and Solaris 9 defaults are insufficient.
`msgtql` (Applies to HP-UX 11i, Solaris 9, and Solaris 10)	Limits the total number of unread messages in the system.	Set to the same value as `msgseg`, that is `32767` or larger.	HP-UX 11i, Solaris 9, and Solaris 10 defaults are insufficient.
`msgmap` (Applies to HP-UX 11i, Solaris 9, and Solaris 10)		Set to the value for `msgtql` plus 2.	Ensure this value remains consistent with `msgtql`.

Enabling WebSphere MQ transaction tracking

After you have installed MQ Tracking, you must enable WebSphere MQ transaction tracking.

About this task

Note: If the target WebSphere MQ infrastructure uses other API exits that modify the data contained in messages, these API exits can potentially interfere with the operation of WebSphere MQ transaction tracking.

Procedure

To enable WebSphere MQ transaction tracking:

1. Ensure that MQ Tracking is installed.
2. Configure the API exits for all queue managers of interest.
3. Deploy the channel message exits to all receiver and requester channels.
4. Restart the queue managers and channels.

What to do next

If you are upgrading your MQ Tracking installation:

1. Stop the queue managers.
2. Upgrade MQ Tracking.
3. Update the API and channel exit definitions.
4. Restart the queue managers and channels.

Configuring API exits

Configure API exits on every queue manager that includes queues that process the interactions between business applications that you want to track.

Before you begin

Important: API exits must be configured on all queue managers of interest, regardless of whether the applications are connecting in client- or bindings-mode.

WebSphere MQ supports a number of methods for configuring API exits; the instructions presented in this guide describe how to use MQ Explorer (on Windows) or the `qm.ini` file (on IBM i, Linux, or UNIX systems) to complete this task. For information about other possible methods of configuring API exits, see the *WebSphere MQ System Administration Guide*.

Regardless of the method you use to configure API exits, the details described in Table 32 are required.

Table 32. Values required to configure API exits

WebSphere MQ installation	Module	Data	Function
32-bit Linux and UNIX queue managers	`/opt/IBM/ITM/`*`arch`*`/th/lib/exits/libTTDCMqExitsServer_`*`version`*	The full path of the directory containing the `ttdcmqexits.cfg` file: `/opt/IBM/ITM/`*`arch`*`/th/config` **Note:** Do not specify the name of the configuration file or the trailing slash in the directory name. The maximum allowed length of this path is 32 characters.	`TTDCMqInitExit`
64-bit Linux and UNIX queue managers	`libTTDCMqExitsServer_`*`version`*		
32-bit Windows queue managers	`C:\IBM\ITM\TMAITM6\kth\exits\TTDCMqExitsServer_`*`version`*`.dll`	The full path of the directory containing the `ttdcmqexits.cfg` file: `C:\IBM\ITM\TMAITM6\kth\config` **Note:** Do not specify the name of the configuration file or the trailing slash in the directory name. The maximum allowed length of this path is 32 characters.	
64-bit Windows queue managers (WebSphere MQ 7.0)	`TTDCMqExitsServer_`*`version`*`.dll`		
IBM i queue managers	`QCYTWMQ/CYTQAXE`	Not required.	

Configuring API exits on Windows systems:

This procedure describes how to configure API exits on Windows using MQ Explorer. On WebSphere MQ 5.3 installations, use the MQServices application.

Procedure

To configure API exits using MQ Explorer:

1. Open the properties for the target queue manager, select the **Exits** page, and click **Add**.
2. Set the following details, using the values listed in Table 32. For example, for Windows x86:
   - **Name**: A name of your choice that describes the exit's purpose, for example `MQTrackingApiExit`

- **Module**: `C:\IBM\ITM\TMAITM6\kth\exits\TTDCMqExitsServer_version.dll`
- **Function**: `TTDCMqInitExit`
- **Data**: `C:\IBM\ITM\TMAITM6\kth\config`

3. Click **OK**, and then close the **Properties** window.

   Note: To use the 64-bit exits on Windows x64 systems, install WebSphere MQ 7.0.0.1 with ifix 7.0.0.1-WS-MQ-Windows-TFP25524 or WebSphere MQ V7.0.1. Ensure the 32-bit exits are located in the default exits directory, and the 64-bit exits are located in the default `exits64` directory of the queue manager that you are configuring. Only specify `TTDCMqExitsServer_version.dll` for the module name. You cannot specify a fully qualified path to the 32-bit or 64-bit exits.

Results

Tip: You can update all queue managers by making these changes to the top-level node in MQ Explorer. If you choose to do this, ensure that you specify the absolute path to the exit module.

What to do next

After configuring an API exit, restart the queue managers.

Tip: Also configure the channel exits before restarting the queue managers.

Configuring API exits on Linux or UNIX systems:

This procedure describes how to configure API exits on Linux or UNIX systems by modifying the queue manager's initialization file.

Before you begin

Queue manager initialization (`qm.ini`) files are located at `/var/mqm/qmgrs/qm_name/qm.ini` where *qm_name* is the name of the queue manager.

Procedure

To configure API exits:

Add the following stanza to the queue manager's `qm.ini` file, using the values listed in Table 32 on page 229. For example, for Linux 32-bit:

```
ApiExitLocal:
   Name=MQTrackingApiExit
   Sequence=next available number divisible by 100
   Module=/opt/IBM/ITM/arch/th/lib/exits/libTTDCMqExitsServer_version
   Function=TTDCMqInitExit
   Data=/opt/IBM/ITM/arch/th/config
```

Note: To use the exits with 64-bit Linux and UNIX queue managers, you can use an unqualified Module name as described in Table 32 on page 229. Ensure that the 32-bit exits are located in the default `exits` directory, and that the 64-bit exits are located in the default `exits64` directory of the queue manager that you are configuring. However, on most UNIX architectures you can instead specify a fully qualified path to the 64-bit exits for the Module name, although doing so can result in the generation of unwanted WebSphere MQ log messages.

Note: Some UNIX architectures include an _r version of the TTDCMqExitsServer module. Do not specify the _r module directly for API exits. WebSphere MQ chooses the appropriate module to use at runtime.

Results

Tip: You can update all queue managers by making these changes to the file /var/mqm/mqs.ini in the ApiExitCommon section. If you choose to do this, ensure that you specify the absolute path to the exit module.

What to do next

After modifying the qm.ini file, restart the queue managers.

Tip: Also configure the channel exits before restarting the queue managers.

Configuring API exits on IBM i systems:

This procedure describes how to configure API exits on IBM i systems by modifying the queue manager's initialization file.

Before you begin

Queue manager initialization files, qm.ini, are located at /QIBM/UserData/mqm/*qm_name*/qm.ini where *qm_name* is the name of the queue manager.

Procedure

To configure API exits:

Add the following stanza to the queue manager's qm.ini file, using the values listed in Table 32 on page 229. For example:

```
ApiExitLocal:
   Name=TTDCMqExits
   Sequence=next available number divisible by 100
   Function=TTDCMqInitExit
   Module=QCYTWMQ/CYTQAXE
```

What to do next

After modifying the qm.ini file, restart the queue managers.

Tip: Also configure the channel exits before restarting the queue managers.

Configuring channel exits

Different channel exits are required to track MQ on distributed systems for different versions of ITCAM for Transactions.

- V7.1 distributed MQ data collector uses SVRCONN channel exits
- V7.1.0.1 distributed MQ data collector does *not* use channel exits
- V7.1.0.2 distributed MQ data collector uses CHADEXIT and CLUSRCVR channel exits
- V7.2 distributed MQ data collector uses CHADEXIT and CLUSRCVR channel exits
- V7.2.0.1 and V7.2.0.2 distributed MQ data collector uses RCVR and RQSTR channel exits only

For ITCAM for Transactions V7.2.0.1 and later, cluster environments for which there is only cluster-related MCA channel activity do *not* require channel exits. Messages sent across cluster channels link horizontally without channel exits. In environments with non-cluster related MCA channel activity, deploy channel message exits to all receiver and requester channels on all queue managers that process interactions between applications that you want to track.

Important: Ensure that you deploy channel message exits to all receiver and requester channels otherwise there will be horizontal link failures in the Transaction Reporter workspaces. If you are upgrading to the V7.2.0.1 MQ data collector, non-cluster channel activity will also fail to link horizontally until you deploy channel message exits to the corresponding receiver and requester channels.

WebSphere MQ supports a number of methods for configuring channel exits. This section describes how to configure channel exits using the **runmqsc** command. For complete information about configuring channel exits, see the *WebSphere MQ System Administration Guide*.

Configuring channel exits on the server:

Use the **runmqsc** command to deploy the channel message exits to all receiver and requester channels.

Before you begin

The following details are required when configuring channel message exits:

Table 33. Values required when deploying channel exits

Platform	Module	Data	Function
Linux and UNIX 32–bit	`/opt/IBM/ITM/`*arch*`/th/lib/exits/libTTDCMqExitsServer_` *version*`_r` **Note:** On Solaris, omit `_r`.	Full path of the directory containing the `ttdcmqexits.cfg` file:`/opt/IBM/ITM/`*arch*`/th/config` **Note:** Do not specify the name of the configuration file or the trailing slash in the directory name. The maximum allowed length of this path is 32 characters.	Channel message exit: `TTDCMqChannelExit`
Linux and UNIX 64–bit	`/opt/IBM/ITM/`*arch*`/th/lib/exits64/libTTDCMqExitsServer_` *version*`_r` **Note:** On Solaris, omit `_r`.		
Windows	`C:\IBM\ITM\TMAITM6\kth\exits\TTDCMqExitsServer_`*version* **Note:** Do not specify the `.dll` extension.	`C:\IBM\ITM\TMAITM6\kth\config` **Note:** Do not specify the name of the configuration file or the trailing slash in the directory name. The maximum allowed length of this path is 32 characters.	
IBM i	`'CYTQCXE_R QCYTWMQ  '` **Note:** There must be exactly one space character between `CYTQCXE_R` and `QCYTWMQ`, and exactly three space characters after `QCYTWMQ`.	Not required.	Not applicable (included with Module).

About this task

Modify the module path, module name, and configuration path to suit the given installation and architecture. Use the 64-bit module path for 64-bit Linux and UNIX queue managers.

Important: Always use reentrant channel exits on the server. All Windows and Solaris exit libraries are reentrant (without the _r suffix). On IBM i, Linux, and other UNIX platforms, reentrant libraries are denoted by _r in the library name.

Procedure

To deploy the channel message exits:

1. On the computer running the queue manager, modify the receiver (RCVR) channel definitions by running the following commands. Substitute the variables for the values listed in Table 33 on page 232.

   For Linux, UNIX, and Windows:

   ```
   > runmqsc qm_name
   > ALTER CHANNEL(channel_name) CHLTYPE(RCVR)
   MSGEXIT('Module(Function)')
   MSGDATA('Data')
   > end
   ```

   For IBM i:

   ```
   > runmqsc qm_name
   > ALTER CHANNEL(channel_name) CHLTYPE(RCVR) MSGEXIT(Module)
   > end
   ```

   For example, for Linux 32-bit:

   ```
   > runmqsc qm_name
   > ALTER CHANNEL(channel_name) CHLTYPE(RCVR)
   MSGEXIT('/opt/IBM/ITM/arch/th/lib/exits/libTTDCMqExitsServer_
   version_r(TTDCMqChannelExit)')
   MSGDATA('/opt/IBM/ITM/arch/th/config')
   > end
   ```

   For example, for IBM i:

   ```
   > runmqsc qm_name
   > ALTER CHANNEL(channel_name) CHLTYPE(RCVR) MSGEXIT('CYTQCXE_R QCYTWMQ   ')
   > end
   ```

2. Repeat step 1 for the requester (RQSTR) channel definitions by specifying CHLTYPE(RQSTR) instead of CHLTYPE(RCVR).
3. For Linux, UNIX, and Windows systems, if you are upgrading to the V7.2.0.1 or later MQ data collector, remove cluster receiver and channel auto definition exits.

What to do next

After configuring the receiver and requester channels, restart the queue managers and start the channels.

Tip: Also configure the API exits before restarting the queue managers.

MQ Tracking configuration file

The configuration file, `ttdcproxy.cfg`, defines operating parameters for MQ Tracking.

Most parameters in this configuration file are set when configuring MQ Tracking using **Manage Tivoli Monitoring Services** or the configuration command for your platform. However, you can directly edit the file. Run the following commands to configure MQ Tracking:

- For IBM i systems, `/QIBM/UserData/cyt/wmq/config/agent.sh config`
- For Linux and UNIX systems, `$CANDLE_HOME/`*`arch`*`/th/config itmcmd config -A th`
- For Windows systems, `%CANDLE_HOME%\TMAITM6\kth\config`

Table 34 describes each parameter.

Table 34. MQ Tracking configuration file

Configuration parameter name	Description
DisablePersistentCaches	Controls the persistence of the buffered transaction data that MQ Tracking collects. Setting this property to `true` means that the buffered data is not persistent, and will not be preserved if MQ Tracking is shut down, which can lead to lost events if MQ Tracking is restarted. Set this value to `false` unless otherwise instructed.
IPCMessageQueueSize	(IBM i, Linux, and UNIX only) Specifies the maximum capacity (in kilobytes) of the System V message queue used by the **IPCMessageQueue** transport. MQ Tracking must be executed with appropriate privileges, such as `root` on Linux and UNIX systems, at least once for this setting to take effect.
IPCQueueDir	Specifies the location of the directory into which MQ transaction data is buffered by API exits and read by MQ Tracking. If the parameter is not defined, it defaults to `$TTDC_SQIPC_DIR`. If `$TTDC_SQIPC_DIR` is also not defined, the effective value of this parameter is `$TEMP` on Windows systems, or `/var/tmp` on Linux, UNIX, and IBM i systems. **Note:** The `IPCQueueDir` parameters in `ttdcmqexits.cfg` and `ttdcproxy.cfg` must match.
IPCQueueLockWaitTime	Sets the timeout (in milliseconds) for each attempt made by MQ Tracking for access to the **IPCSharedQueue**. A value of `0` indicates no time out.
LogLevel	Sets the amount of log information written to the MQ Tracking log file; set this property as required. The available levels are `error`, `warning`, `info`, and `debug`. For Linux, UNIX, and Windows systems, the log file is located in the IBM Tivoli Monitoring logs directory. For IBM i systems, the log file is located at `/QSYS.LIB/QCYTWMQ.LIB/LOG.FILE`.
QueueIdleDelay	Specifies the delay (in milliseconds) between checks by MQ Tracking for queue entries when there is nothing to retrieve.
TTConnectionRetryInterval	Specifies the interval (in milliseconds) between attempts to connect to the Transaction Collector if a connection attempt fails.
TTConnectRetries	Specifies the number of attempts made by the MQ Tracking to connect to the Transaction Collector. A value of 0 indicates unlimited retries.
TTConnectTimeout	Specifies the timeout (in milliseconds) for each attempt made by the MQ Tracking to connect to the Transaction Collector.

Table 34. MQ Tracking configuration file (continued)

Configuration parameter name	Description
TTServerString	Specifies the protocol, IP address, and port that MQ Tracking uses to send tracking events to the Transaction Collector. For IPv4, use the format *protocol*:*IPv4 address*:*port*, for example tcp:127.0.0.1:5455. For IPv6, use the format *protocol*:[*IPv6 address*]:*port*, for example, tcp:[::1]:5455 is equivalent to the previous IPv4 example. The only protocol currently supported is TCP.

Exit configuration file

The exit configuration file, ttdcmqexits.cfg, defines operating parameters for the MQ transaction tracking exits.

Most parameters in this configuration file are set when configuring MQ Tracking using **Manage Tivoli Monitoring Services** or the configuration command for your platform. However, you can directly edit the file. Run the following commands to configure MQ Tracking:

- For IBM i systems, /QIBM/UserData/cyt/wmq/config/agent.sh config
- For Linux and UNIX systems, $CANDLE_HOME/*arch*/th/config itmcmd config -A th
- For Windows systems, %CANDLE_HOME%\TMAITM6\kth\config

Table 35 describes each parameter.

Table 35. API exit configuration file

Configuration parameter name	Description
DefaultQmgr	Specifies the name of the default queue manager. If nothing is specified for this parameter, it is assumed that there is no default queue manager on the system. When the MQ data collector is configured, this parameter is automatically set to the name of the default MQ queue manager for this host.
ExitProcFilter	Specifies a comma-separated list of filter strings that the API exit applies to the command line of any process initializing it. If the process's command line contains any of the filter strings, the API exit does not register itself with that process. Use this parameter to suppress tracking of messages sent by particular applications. For example: ExitProcFilter = AMQPCSEA,amqpcsea,com.ibm.mq.explorer.ui.rcp.RcpApplication,kmcrca,mmc.exe See "Filtering WebSphere MQ transactions" on page 237 for more information.
IPCPrimaryTransport	Specifies which IPC transport the exits will use, either the **IPCSharedQueue** or the **IPCMessageQueue**. This setting is only relevant to Linux, UNIX, and IBM i systems; Windows systems always use **IPCSharedQueue**.
IPCQueueDir	Specifies the location of the directory into which MQ transaction data is buffered by API exits and read by MQ Tracking. If not otherwise defined, the parameter is set to *value of MQTrackingInstallRoot* on Windows, Linux, and UNIX systems, or /QIBM/UserData/cyt/wmq/queues on IBM i systems. **Note:** The IPCQueueDir parameters in ttdcmqexits.cfg and ttdcproxy.cfg must match.
IPCQueueLockWaitTime	Sets the timeout (in milliseconds) for each attempt made by the exits for access to the **IPCSharedQueue**. A value of 0 indicates no time out.

Table 35. API exit configuration file (continued)

Configuration parameter name	Description
LogDir	Specifies the directory in which API exits and channel exits place their log files. If this parameter is not set, the exit places it in the following locations: • On IBM i systems, `/QIBM/UserData/cyt/wmq/config/logs` • On Linux and UNIX systems, `$CANDLE_HOME/logs` • On Windows systems, `$SYSTEMDRIVE\IBM\ITM\logs` Log files have the name *hostname*`_th_mqexits.log`.
LogLevel	Sets the amount of log information written to the exit log file by API exits and channel exits. The available levels are `error`, `warning`, `info`, and `debug`. The default level for V7.2.0.1 and later is `Warning`. Where you have upgraded from V7.2.0.0 or earlier, update **LogLevel** in `ttdcmqexits.cfg` to `Warning` to display additional IPC-related diagnostic messages.
LogLimit	Sets the maximum log file size that can be reached before the log is truncated. A value of **0** indicates that the size of the log file is unlimited and can grow indefinitely. The maximum value that you can specify is **2048** MB. The default value is **10** MB.
MQTrackingInstallRoot	(Linux, UNIX, and Windows systems only) The base directory to which the MQ Tracking files are installed. This is automatically configured during installation and should not be changed.
QueueFilterExclusions	Specifies a comma-separated list of filter strings that are used by the API exits to decide when activity on a queue should not be exempt from tracking. If a queue name contains any of the filter strings, the API exit will always track activity on that queue. Use this parameter to enable tracking of messages through queues that would otherwise be excluded. For example: `QueueFilterExclusions = SYSTEM.CLUSTER.TRANSMIT.QUEUE` By default, all `SYSTEM` queues other than `SYSTEM.CLUSTER.TRANSMIT.QUEUE` are excluded from tracking.
QmgrSyncpointFilter	Specifies a comma-separated list of queue manager names that the API exits use to decide whether or not typical syncpoint behavior should be followed. If the name of a queue manager exactly matches any of the filter strings or if any of the filter strings is the single wildcard character '*', then all syncpoint activity on that queue manager is assumed to be committed. Backed out queue activity will not be handled accurately. For example: `QmgrSyncpointFilter = QMA,QMB` By default, this filter is empty.
SharedBuildNumber	(Linux, UNIX, and Windows systems only) The shared library version when interoperability with other transaction tracking domains, such as Data Collector for WebSphere Message Broker, is required. This is automatically configured during installation and should not be changed.

Filtering WebSphere MQ transactions

MQ Tracking permits you to filter out transactions in the WebSphere MQ domain. This enables you to ignore transactions made by particular applications that are not of interest.

Before you begin

The exit configuration file, `ttdcmqexits.cfg`, provides the `ExitProcFilter` parameter, which specifies the processes whose transactions are filtered out. The parameter contains a comma-separated list of filter strings, each representing the command line of processes whose WebSphere MQ transactions are not tracked.

When a message exchange occurs, the WebSphere MQ transaction tracking exit applies these filters to the command line of the process making the exchange. If the command line matches one or more of the filters, the exchange is not tracked and no transaction is reported. Exits apply filter strings in a case-sensitive substring match.

Procedure

To filter out transactions made by a particular application:

Open the `ttdcmqexits.cfg` file in a text editor and append to the `ExitProcFilter` parameter a filter string matching the command line used to launch that application.

Example

Important: To avoid unintentional filtering of transactions, choose a filter that matches the executable or class name of the target process as closely as possible. For example, use the string `com.ibm.mq.explorer.ui.rcp.RcpApplication` in preference to `com`.

What to do next

By default, transactions from the following applications are filtered:

- WebSphere MQ Explorer
- WebSphere MQ Command server
- OMEGAMON® XE for Messaging configuration agent
- Microsoft Management Console

Uninstalling MQ Tracking

Uninstall MQ Tracking from each computer to which it is installed.

Procedure

To uninstall MQ Tracking on all platforms:

1. Unconfigure the API exits and channel exits.
2. Stop or restart the queue managers.
3. Uninstall MQ Tracking. See “Uninstalling Transaction Tracking” on page 210 for information about how to do this.

Preparing CICS Tracking

CICS Tracking monitors transactions as they flow through CICS components on z/OS systems and generates tracking data about those transactions.

CICS Tracking is available in ITCAM for Transactions V7.1 and later.

Table 36 lists the intra-domain and inter-domain tracking supported by CICS Tracking.

Table 36. CICS Tracking - domain interactions supported

Domain interactions tracked	Notes
CICS to and from CICS	SOAP, DPL, MRO and IPIC, are supported.
WebSphere Application Server to and from CICS	Through ECI.
WebSphere MQ to and from CICS	EXCI, Sync on Return, and XA supported
CICS TG to and from CICS	
DB2 to and from CICS	

For further information about CICS Tracking, see CICS Tracking in the IBM Tivoli Composite Application Manager for Transactions Installation and Configuration Guide for z/OS.

Preparing 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 is available in ITCAM for Transactions V7.1.0.1 and later.

CICS TG Transaction Tracking consists of CICS TG Transaction Tracking monitoring exits for CICS TG Client Application Classes and for CICS TG Gateway Daemon.

Table 37 lists the inter-domain tracking supported by CICS TG Transaction Tracking.

Table 37. CICS TG Transaction Tracking - domain interactions supported

Domain interactions tracked	Notes
WebSphere Application Server to and from CICS TG	Through CICS.
Remote client to CICS TG	EXCI, Sync on Return, ECI extended, and XA supported.
CICS TG to CICS	EXCI, Sync on Return, and XA supported

To prepare CICS TG transaction tracking:

1. Install CICS TG Transaction Tracking monitoring exits on all distributed systems running CICS TG Client Application Classes whose transactions you want to track.

2. Install CICS TG Transaction Tracking monitoring exits on the CICS TG Gateway Daemon on z/OS and configure the daemon to use the monitoring exits. CICS TG Transaction Tracking monitoring exits for the daemon are installed with Transactions Base, but must be configured manually.
3. If monitoring CICS TG transaction flows from a WebSphere Application Server Java 2 Platform, Enterprise Edition (J2EE) server, customize the J2EE Connector Architecture (JCA) connections to use CICS TG Transaction Tracking monitoring exits.
4. If using CICS TG Transaction Tracking with stand-alone applications that use the CICS TG JavaGateway Client Class, customize the application code to use the CICS TG Transaction Tracking monitoring exits.
5. Configure CICS TG Transaction Tracking using the configuration file for each CICS TG Client Application Classes and each CICS TG Gateway Daemon.

Prerequisites

Before installing CICS TG Transaction Tracking, ensure that the following applications are installed and communicating:

- WebSphere Application Server or a supported stand-alone application
- CICS TG
- CICS

CICS TG Transaction Tracking supports transaction tracking on:

- CICS TG Client Application Classes version 7.1 and later on distributed systems
- CICS TG Gateway Daemon version 7.1 and later on z/OS

If you are using CICS TG resource adapter files on WebSphere Application Server the following are required:

- CICS TG resource adapter files 7.1 and later
- WebSphere Application Server version 6.1
- Java Virtual Machine V5.0

If you are using the ITCAM for Application Diagnostics (was ITCAM for WebSphere) FP3 IF15 data collector with CICS TG Transaction Tracking's data collector, both data collectors must be configured to send Transaction Tracking API events to the same Transaction Collector.

Limitations

Flows are not correlated between the gateway daemon and CICS if a user replaceable module (DFHXCURM) is used that re-routes EXCI requests to a CICS region other than the region to which the original request would have flowed.

CICS TG Transaction Tracking monitoring exits support synchronous flows through the CICS TG Gateway Daemon where the gateway daemon is using EXCI to invoke CICS programs. IPIC communication is not tracked between the gateway daemon and CICS.

CICS TG Transaction Tracking monitoring exits support tracking ECI SYNCONRETUN and XA flow types between the CICS TG Gateway Daemon and CICS. ECI Extended LUW mode flows between the CICS TG Gateway Daemon and CICS are not correlated.

Installing CICS TG Transaction Tracking on Linux, UNIX, and Windows systems

Install CICS TG Transaction Tracking monitoring exits on all computers running CICS TG Client Application Classes with transactions that you want to track.

Before you begin

Ensure that you have read and understood the prerequisites described in "Preparing CICS TG Transaction Tracking" on page 238.

About this task

The CICS TG Transaction Tracking installer is archived with Transaction Tracking. For information about obtaining the latest version, see the Download information on the ITCAM for Transactions Information Center.

Procedure

To install CICS TG Transaction Tracking:

1. Extract the CICS TG Transaction Tracking installer, `ctg-ttmonitoring-`*`platform`*`-[32|64]-v`*`buildnumber`*, from the Transaction Tracking DVD. For example, on Windows systems, `ctg-ttmonitoring-win-32-v2088.exe`.
2. Run `ctg-ttmonitoring-`*`platform`*`-[32|64]-v`*`buildnumber`* as an administrator.
3. On the **Welcome** window, click **Next**.
4. Enter the path to the directory where the files will be installed. For example, `C:\IBM\ITM\TMAITM6\ktj\ctg` on Windows systems, or `/opt/IBM/ITM/`*`os`*`/tj/ctg` on UNIX and Linux systems. This directory is referred to as *ctgt.install.dir* in this documentation.
5. Click **Next** to begin the installation.

   The files are extracted to *ctgt.install.dir*.
6. Check for errors and click **Finish** to complete the installation.

What to do next

Install CICS TG Transaction Tracking on CICS TG Gateway Daemon on z/OS and then configure the required elements so that CICS TG Transaction Tracking can interact with your system.

Installing and configuring CICS TG Transaction Tracking on z/OS

Install CICS TG Transaction Tracking on the CICS TG Gateway Daemon on z/OS.

About this task

CICS TG Transaction Tracking is installed with Transactions Base V7.3. See CICS TG Transaction Tracking in the *Installation and Configuration Guide for z/OS* for further information.

What to do next

Configure the CICS TG Gateway Daemon on z/OS to use CICS TG Transaction Tracking. See CICS TG Transaction Tracking in the *Installation and Configuration Guide for z/OS* for further information.

Customizing WebSphere Application Server for CICS TG Transaction Tracking

To monitor CICS TG transactions from a WebSphere Application Server Java 2 Platform, Enterprise Edition (J2EE) server, customize the J2EE Connector Architecture (JCA) connections to use CICS TG Transaction Tracking monitoring exits.

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 JCA connection factories that have been created to use the listed CICS TG 7.1 resource adaptors. These connection factories must be individually configured to use the CICS TG Transaction Tracking monitoring exits.

Procedure

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

1. Add `ctg-tt.jar` to the **`classpath`** for CICS TG resource adaptors 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/ctg-tt.jar
      ctgt.install.dir/lib/ttdc-jni.jar
      ctgt.install.dir/lib/ttapi4j.jar
      ```

   d. Add the following text to the **Native path** field in WebSphere Application Server:

      ```
      ctgt.install.dir/lib
      ```

   e. Repeat this step for the other resource adaptor if both are installed.
2. Set each monitored JCA 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 JCA 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 JCA 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.
   f. Repeat this step for all the CICS TG JCA connection factories that you want to monitor.
3. Restart WebSphere Application Server.

What to do next

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

To enable stitching between WebSphere Application Server and CICS TG Transaction Tracking 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 for stand-alone applications

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

About this task

To enable CICS TG Transaction Tracking monitoring exits in the client application, set the following JVM property:

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

Alternatively, enter the following code and recompile the client application to enable CICS TG Transaction Tracking monitoring exits:

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

Note: In either case, the `ctg-tt.jar` file must be on the classpath of the application for the monitoring exit to be found. The `ctgt.inst.dir/lib` folder must also be on the native path or library path of the application.

What to do next

Configure the CICS TG Transaction Tracking monitoring exits in the configuration file.

Configuring CICS TG Transaction Tracking

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.

Before you begin

Ensure that CICS TG Transaction Tracking is installed.

About this task

The CICS TG Transaction Tracking configuration file contains a number of parameters that affect the behavior of the CICS TG Transaction Tracking monitoring exits, including the Transaction Collector address, DNS options, CICS region for z/OS, and log settings. See the example `ttdcctgexits.cfg` file in the installation directory for further information.

The location and name of the configuration file used by CICS TG Transaction Tracking depends on the operating system:

- On distributed platforms, `ttdcctgexits.cfg` installed in the same directory as `ctg-tt.jar`
- On z/OS systems, `cytgexitconfig.cfg` installed in the same directory as `cytgtt.jar`

Procedure

To edit the CICS TG Transaction Tracking configuration file:

1. Open `ttdcctgexits.cfg` or `cytgexitconfig.cfg` with a text editor.
2. Edit the required parameters.
3. Save and close the configuration file.
4. Restart any CICS TG Gateway Daemon, WebSphere Application Server or stand-alone client application that employs CICS TG Transaction Tracking using this configuration file.

What to do next

To specify a different configuration file for CICS TG Transaction Tracking if required:

1. Set the Java system property `com.ibm.transactions.ctg.config` to point to the alternate configuration file.

   For example, on the Java command line:

   ```
   java -Dcom.ibm.transactions.ctg.config=/var/ctgalternative.cfg
   ```

2. Alternately, if you are using CICS TG Transaction Tracking monitoring exits with WebSphere Application Server, you can set this property as part of the startup scripts.

Preparing IMS Tracking and IMS Connect Tracking

IMS Tracking monitors transactions as they flow through IMS components on z/OS systems and generates tracking data about those transactions.

IMS Tracking is available in ITCAM for Transactions V7.1 and later.

Table 38 lists the intra-domain and inter-domain tracking supported by IMS Tracking.

Table 38. IMS Tracking - domain interactions supported

Domain interactions tracked	Notes
IMS to and from IMS	Shared queues and MCS are supported.
WebSphere Application Server to and from IMS	Through IMS Connect.
WebSphere MQ to and from IMS	
DB2 to and from IMS	Using IMS Tracking with ITCAM for Transactions V7.1.0.2 and later.
IMS Connect to and from IMS	Using IMS Connect Tracking with ITCAM for Transactions V7.2 and later.

For further information about IMS Tracking, see IMS and IMS Connect Tracking in the Installation and Configuration Guide for z/OS.

Preparing Data Collector for WebSphere Message Broker

Data Collector for WebSphere Message Broker tracks messages passing through WebSphere Message Broker nodes.

The Data Collector for WebSphere Message Broker is a shared IBM Tivoli Monitoring component, which works with both ITCAM for Transactions and ITCAM for SOA. The Data Collector for WebSphere Message Broker supersedes WebSphere Message Broker Tracking in ITCAM for Transactions V7.3 and later.

There are no specific prerequisites for installing Data Collector for WebSphere Message Broker. To stitch transactions between the WebSphere Message Broker and the WebSphere MQ domains, install MQ Tracking on the same computer, and point the **ttdc.mq.InstallDir** parameter in the Message Broker Tracking configuration file to the MQ Tracking installation directory.

Table 39 lists the inter-domain tracking supported by Data Collector for WebSphere Message Broker.

Table 39. Data Collector for WebSphere Message Broker - domain interactions supported

Domain interactions tracked	Notes
WebSphere Message Broker to and from WebSphere MQ	Requires MQ Tracking
WebSphere Message Broker to and from Microsoft .NET	SOAP only Requires .NET Data Collector
WebSphere Message Broker to IBM HTTP Server	HTTP only Requires ARM

Table 39. Data Collector for WebSphere Message Broker - domain interactions supported (continued)

Domain interactions tracked	Notes
WebSphere Message Broker to and from WebSphere Application Server	SOAP only Requires ITCAM for Application Diagnostics

Installing Data Collector for WebSphere Message Broker on Windows, Linux, and UNIX systems

Install Data Collector for WebSphere Message Broker on each computer running a WebSphere Message Broker that you want to monitor.

Before you begin

For information about obtaining the latest version of Transaction Tracking, see the Download information in the required ITCAM for Transactions Information Center.

Procedure

To install Data Collector for WebSphere Message Broker on Windows, Linux, and UNIX systems:

1. Launch the Transaction Tracking installer.
2. Install the following features:
   - Tivoli Enterprise Monitoring Agent Framework
   - Data Collector for WebSphere Message Broker

What to do next

Enable Data Collector for WebSphere Message Broker using **configDC**. See "Enabling the Data Collector for WebSphere Message Broker" for further information.

Note: Do not use `kinconfg.exe` on Windows or `itmcmd config` on UNIX systems to configure Data Collector for WebSphere Message Broker.

Enabling the Data Collector for WebSphere Message Broker

After you have installed the Data Collector for WebSphere Message Broker, you must enable it.

Before you begin

Before enabling the Data Collector for WebSphere Message Broker, you should first configure the `KK3.dc.properties` file. Set the following properties:

- `default.monitor=on`
- `default.tt.enabled=true`
- `default.tt.serverstring=`*`address of Transaction Collector`*
- `default.ttdc.mq.installdir=`*`path-to-MQTracking-installation-directory`*

See "KK3.dc.properties" on page 515 for further information.

About this task

To enable the Data Collector for WebSphere Message Broker, enable the KK3UserExit for all Message Flows to be monitored. The KK3UserExit can be enabled for individual Message Flows, or it can be enabled for an entire WebSphere Message Broker instance.

Procedure

To enable the Data Collector for WebSphere Message Broker environment:

1. Run the `configDC` script to configure an environment to load the user exit library. For example:

   ```
   cd /opt/IBM/ITM/aix513/k3/bin
   ./configDC.sh –enable /opt/IBM/mqsi/7.0
   ```

2. Close the consoles or shells:
   - On Windows systems, close all WebSphere Message Broker Command Consoles
   - On UNIX and Linux systems, close all shells that have loaded the MQSI environment
3. Load the `MQSI` environment:

   On Windows systems, select the **Command Console** shortcut from the WebSphere Message Broker start menu.

   On UNIX systems, use the `mqsiprofile` script in the WebSphere Message Broker install bin directory. For example, `. /opt/ibm/mqsi/7.0/bin/mqsiprofile`.
4. Stop the WebSphere Message Broker instance on which you want to configure user exits.

   ```
   mqsistop broker_name
   ```

5. Optional: Run the `mqsichangebroker` command to enable the KK3UserExit for all Message Flows on the WebSphere Message Broker instance:

   ```
   mqsichangebroker broker_name -e "KK3UserExit"
   ```

   If this step is skipped, the KK3UserExit is disabled for all Message Flows by default.

   Note: This command overrides the current set of default active user exits. Specify additional user exits using the colon character.
6. Restart the message brokers to load the user exit libraries. For example:

   ```
   mqsistart broker_name
   ```

7. Optional: If you skipped step 5, run the following command to enable the KK3UserExit for a particular Message Flow:

   ```
   mqsichangeflowuserexits broker_name -e egName -f \
      messageFlowName -a "KK3UserExit"
   ```

   Tip: Specify additional user exits using a colon-separated list.

Upgrading from WebSphere Message Broker Tracking to the Data Collector for WebSphere Message Broker

Follow these steps to upgrade from WebSphere Message Broker Tracking, included in V7.2.0.2 and earlier, to the Data Collector for WebSphere Message Broker included in V7.3 and later.

About this task

Attention: Any brokers or flows that are still configured to use `TTDCUserExit` after you uninstall WebSphere Message Broker Tracking will not start. Ensure that no brokers or flows are configured to use `TTDCUserExit` before you uninstall WebSphere Message Broker Tracking. The user exits for a broker or flow can be verified using the **mqsireportflowuserexits** command

Procedure

To upgrade from WebSphere Message Broker Tracking to the Data Collector for WebSphere Message Broker:

1. Install the Data Collector for WebSphere Message Broker.
2. Disable `TTDCUserExit` for all message brokers and flows.
3. Uninstall WebSphere Message Broker Tracking.
4. Enable the WebSphere Message Broker environment. See "Enabling the Data Collector for WebSphere Message Broker" on page 245 for further information.

Upgrading from ITCAM for SOA Message Broker data collector to the Data Collector for WebSphere Message Broker

Follow these steps to upgrade from ITCAM for SOA Message Broker data collector, included in ITCAM for SOA V7.1.1.1 and earlier, to the Data Collector for WebSphere Message Broker included in ITCAM for Transactions V7.3 and later.

Procedure

To upgrade from ITCAM for SOA Message Broker data collector to the Data Collector for WebSphere Message Broker:

1. Install the Data Collector for WebSphere Message Broker.
2. Disable the ITCAM for SOA Message Broker data collector for all flows using either the ITCAM for SOA Data Collector Configuration Utility or **KD4configDC** script.
3. Enable the WebSphere Message Broker environment. See "Enabling the Data Collector for WebSphere Message Broker" on page 245 for further information.

Uninstalling the Data Collector for WebSphere Message Broker

Before uninstalling the Data Collector for WebSphere Message Broker, first remove the KK3UserExit from all WebSphere Message Broker instances and Message Flows.

Procedure

To uninstall the Data Collector for WebSphere Message Broker:

1. Load the `MQSI` environment:

On Windows systems, select the **Command Console** shortcut from the WebSphere Message Broker start menu.

On UNIX systems, use the `mqsiprofile` script in the WebSphere Message Broker install bin directory. For example, `. /opt/ibm/mqsi/7.0/bin/mqsiprofile`.

2. Optional: Run the following command to disable the KK3UserExit for a particular Message Flow:

```
mqsichangeflowuserexits broker_name -e egName -f \
   messageFlowName -a ""
```

3. Stop the WebSphere Message Broker instance from which you want to remove user exits.

```
mqsistop broker_name
```

4. Optional: If you skipped step 2, run the `mqsichangebroker` command to disable the KK3UserExit for all Message Flows on the WebSphere Message Broker instance:

```
mqsichangebroker broker_name -e ""
```

Note: This command overrides the current set of default active user exits.

5. Run the `configDC` script to remove the user exit library. For example:

```
cd /opt/IBM/ITM/aix513/k3/bin
./configDC.sh –disable /opt/IBM/mqsi/7.0
```

6. Run the Transaction Tracking installer and uninstall the `Data Collector for WebSphere Message Broker` Tivoli Enterprise Management Agent.
7. Restart the message brokers to remove the user exit libraries. For example:

```
mqsistart broker_name
```

Preparing your .NET tracking environment

To prepare .NET transaction tracking, install and configure the .NET Data Collector on each computer running .NET whose applications you want to monitor.

.NET Data Collector is available in V7.3 and later. .NET Data Collector replaces .NET Tracking available in ITCAM for Transactions V7.2, V7.2.0.1, and V7.2.0.2.

Table 40. Services monitored by .NET Tracking and .NET Data Collector

.NET Tracking in ITCAM for Transactions V7.2, V7.2.0.1, V7.2.0.2	**.NET Data Collector in ITCAM for Transactions V7.3**
ASMX/WCF web services	ASMX/WCF web services
	ADO.NET (DB2, ODBC, OleDB, Oracle, SQL) at the client side
	Directory service interfaces (ADSI, LDAP) at the client side
	IIS Server for web applications and basic authentication

Table 41 on page 249 lists the intra-domain tracking supported by .NET Data Collector.

Table 41. .NET Data Collector - domain interactions supported

Domain interactions tracked	Notes
.NET to IBM HTTP Server	WCF applications link to IBM HTTP Server, where IBM HTTP Server is monitored by ARM.
.NET to WebSphere Application Server	SOA .NET applications can interact with web services running on WebSphere Application Server where the web services are monitored by ITCAM for Application Diagnostics.
.NET to and from WebSphere Message Broker	WCF/ASMX web services.

.NET Data Collector architecture

.NET Data Collector consists of several monitoring components which together provide an integrated view of the .NET transactions flowing through the monitoring points described here. The following components are installed during the installation of .NET Data Collector and registered using the same command:

- .NET profiler - uses the .NET CLR Profiler to provide ADO.NET database and directory service client side transaction tracking

  See Appendix I, "ADO.NET supported namespaces," on page 517 for a list of supported namespaces.
- SOAP handlers - provides SOAP handlers to monitor ASMX and WCF requests and responses
- Internet Information Services (IIS) ISAPI filter - monitors HTTP requests and responses, and basic IIS authentication transactions

Preparing your environment

To prepare your environment to monitor .NET applications:

1. Identify all computers running applications on .NET that you want to monitor.
2. On each of these computers, install .NET Data Collector.
3. When installation is complete, configure .NET Data Collector.

Prerequisites

Before installing .NET Data Collector, ensure that the following applications are installed and communicating:

- Windows operating system support
- .NET Framework
- ITCAM for Transactions V7.3 or later
- IIS Server

See "ITCAM for Transactions supported software" for information about how to find the latest supported versions of the prerequisite software.

ITCAM for Transactions supported software

For information about supported software:

1. Link to the required ITCAM for Transactions version from ITCAM for Transactions on Documentation Central
2. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**.
3. In the System requirements and prerequisites page, select Transaction Tracking from the Compatible software list.

Installing .NET Data Collector

Install .NET Data Collector on all computers running .NET whose transactions you want to track.

Before you begin

.NET Data Collector is included in the Transaction Tracking V7.3 and later package and is installed using a separate installer. The installer is named `dotnet-ttmonitoring-`*`arch-bit-buildnumber`*. For example, `dotnet-ttmonitoring-win32-32-v7300.exe`. The 32-bit installer can also be used on 64-bit systems.

For information about obtaining the latest version, see the Download information on the ITCAM for Transactions Information Center.

Ensure that you have read and understood the prerequisites information described in "Preparing your .NET tracking environment" on page 248.

About this task

The following procedure describes the installation of .NET Data Collector on Windows systems.

Note: .NET Data Collector can be installed only once on your system.

Procedure

To install .NET Data Collector on Windows systems:

1. Copy the .NET Data Collector installer, `dotnet-ttmonitoring-`*`arch-bit-buildnumber`*, from the Transaction Tracking DVD.
2. Stop all .NET applications and services.
3. Run the .NET Data Collector installer as an administrator.
4. On the **Welcome** window, click **Next**.
5. On the **Installation path** window, set the directory to which the .NET Data Collector files will be installed, and click **Next**:
   - For new installations, specify the path to the required directory. For example, `C:\IBM\ITM\TMAITM6\ktj\dotnet`.
   - For upgrades, if a previous .NET Tracking installation is detected that installation directory is shown as the default. You cannot change this directory.

   This directory is referred to as *dotnet.install.dir* in this documentation.
6. On the **Transaction Collector host/IP address** window, set the host name of the Transaction Collector, and click **Next**.
7. On the **Transaction Collector port** window, set the port for communication with the Transaction Collector, and click **Next**.

8. On the **Monitor .NET transactions immediately** window, select **Yes**, and click **Next**.

   Setting this option runs the **configdc registerdc** command. The command registers the .NET profiler, ASMX/WCF SOAP handlers, and the IIS ISAPI filter if IIS is detected. By default, all supported .NET interfaces are enabled for monitoring.

   If you prefer to start monitoring later, select **No** and manually run the **configdc registerdc** command when required. See "Configuring registration" on page 254 for further information.
9. When the installation completes, click **Finish**.

What to do next

Configure .NET Data Collector. See "Configuring .NET Tracking" on page 253.

Installing .NET Data Collector with an existing .NET Tracking data collector

You can install .NET Data Collector onto a system that already has the .NET Tracking data collector installed.

Before you begin

.NET Data Collector is included in the Transaction Tracking V7.3 and later package and is installed using a separate installer. The installer is named `dotnet-ttmonitoring-arch-bit-buildnumber`. For example, `dotnet-ttmonitoring-win32-32-v7300.exe`. The 32-bit installer can also be used on 64-bit systems.

For information about obtaining the latest version, see the Download information on the ITCAM for Transactions Information Center.

Ensure that you have read and understood the prerequisites information described in "Preparing your .NET tracking environment" on page 248.

About this task

The following procedure describes how to install the .NET Data Collector on Windows systems.

Note: .NET Data Collector can be installed only once on your system.

Procedure

1. Stop all .NET applications and services.
2. Run the .NET Data Collector installer as an administrator.
3. On the **Welcome** window, click **Next**.
4. On the **Installation path** window, set the directory to which the .NET Data Collector files will be installed. For example, `C:\IBM\ITM\TMAITM6\ktj\dotnet`. This directory is referred to as *dotnet.install.dir* in this documentation. Do not install the .NET Data Collector into the .NET Tracking installation directory, instead specify a different installation directory for the .NET Data Collector.
5. On the **Transaction Collector host/IP address** window, set the host name of the Transaction Collector, and click **Next**.
6. On the **Transaction Collector port** window, set the port for communication with the Transaction Collector, and click **Next**.

7. On the **Monitor .NET transactions immediately** window, select **Yes**, and click **Next**.

   Setting this option runs the **configdc registerdc** command. The command registers the .NET profiler, ASMX/WCF SOAP handlers, and the IIS ISAPI filter if IIS is detected. By default, all supported .NET interfaces are enabled for monitoring.

   If you prefer to start monitoring later, select **No** and manually run the **configdc registerdc** command when required. See "Configuring registration" on page 254 for further information.
8. When the installation completes, click **Finish**.

What to do next

Configure .NET Data Collector. See "Configuring .NET Tracking" on page 253.

Installing .NET Data Collector with an existing ITCAM for SOA .NET data collector

You can install .NET Data Collector onto a system that already has the ITCAM for SOA agent installed and .NET monitoring enabled. Replace the existing ITCAM for SOA .NET data collector with the .NET Data Collector to display both transaction information about ASMX/WCF, ADO.NET, ISAPI, and web services metrics in the ITCAM for SOA agent or ITCAM for Transactions workspaces.

Before you begin

.NET Data Collector is included in the Transaction Tracking V7.3 and later package and is installed using a separate installer. The installer is named `dotnet-ttmonitoring-arch-bit-buildnumber`. For example, `dotnet-ttmonitoring-win32-32-v7300.exe`. The 32-bit installer can also be used on 64-bit systems.

For information about obtaining the latest version, see the Download information on the ITCAM for Transactions Information Center.

Ensure that you have read and understood the prerequisites information described in "Preparing your .NET tracking environment" on page 248.

About this task

.NET Data Collector can be installed only once on your system.

The existing ITCAM for SOA .NET data collector Transaction Tracking server configuration is not migrated to .NET Data Collector. The ASMX/WCF handlers use the Transaction Tracking server configuration provided during .NET Data Collector installation or by the `configdc ttserver` command.

After .NET Data Collector is installed and configured:

- Only the ASMX/WCF handlers relate to the SOA agent. The .NET profiler and ISAPI are not affected.
- The ASMX/WCF handlers obtain their configuration from the ITCAM for SOA .NET installation directory, but use their own libraries in the .NET Data Collector installation directory: `kd4dcagent`, `kd4soapagent`, and `kd4wcfagent`.
- The SOA agent workspace displays the real time services messages for ASMX/WCF monitored by .NET Data Collector.
- ITCAM for SOA .NET Take Action commands are supported.

- Trace logs from the ASMX/WCF handlers for .NET Data Collector are saved to the .NET Data Collector installation directory, `ktj/dotnet/logs`, not to the ITCAM for SOA .NET installation directory. Logging for ASMX/WCF handlers is controlled using ITCAM for SOA Take Action commands.

Procedure

1. Stop all .NET applications and services.
2. Run the .NET Data Collector installer as an administrator.
3. On the **Welcome** window, click **Next**.
4. On the **Installation path** window, set the directory to which the .NET Data Collector files will be installed, and click **Next**. For new installations, specify the path to the required directory. For example, `C:\IBM\ITM\TMAITM6\ktj\dotnet`. This directory is referred to as *dotnet.install.dir* in this documentation.
5. On the **Transaction Collector host/IP address** window, set the host name of the Transaction Collector, and click **Next**.
6. On the **Transaction Collector port** window, set the port for communication with the Transaction Collector, and click **Next**.
7. On the **Monitor .NET transactions immediately** window, select **Yes**, and click **Next**.

   Setting this option runs the **`configdc registerdc`** command. The command registers the .NET profiler, ASMX/WCF SOAP handlers, and the IIS ISAPI filter if IIS is detected. By default, all supported .NET interfaces are enabled for monitoring.

   If you prefer to start monitoring later, select **No** and manually run the **`configdc registerdc`** command when required. See "Configuring registration" on page 254 for further information.
8. When the installation completes, click **Finish**.

What to do next

Edit the `dotnet.install.dir\config\dotNetDcConfig.properties.inactive` file and set the following parameter: `default.kd4.enabled=true`.

Configure .NET Data Collector. See "Configuring .NET Tracking."

Configuring .NET Tracking

Configure .NET Data Collector after you have installed it.

After you have installed .NET Data Collector, complete the following configuration steps:

1. Registration
2. Transaction Collector connectivity
3. Enable monitoring of specific ADO.NET interfaces
4. Activate the configuration

Configuring registration

Configure the Windows environment so that .NET Data Collector can intercept .NET transactions.

About this task

This task registers all three .NET Data Collector components to enable .NET transaction monitoring.

Remember: Reregister and reactivate the .NET Data Collector whenever you update the .Net framework.

Procedure

To configure registration:

1. Register the .NET Data Collector profiling DLL by running the following command:

   ```
   configdc registerdc [-f]
   ```

   where: `-f` overrides any existing .NET profiler without further prompts. If a profiler exists and the `-f` option is not specified, the .NET Data Collector profiler is not configured.
2. Restart all running .NET applications.

What to do next

After you have configured registration, configure connection to the Transaction Collector.

If you want to stop .NET Data Collector intercepting .NET transactions, run the following command and then restart the .NET applications:

```
configdc unregisterdc
```

Configuring connection to the Transaction Collector

Configure connection to the Transaction Collector after you have installed .NET Data Collector. .NET Data Collector events are sent to this server.

About this task

The location of the Transaction Collector should have been set in the configuration file automatically during installation. If you were not prompted for the location of the Transaction Collector during installation of .NET Data Collector, you may need to do this manually.

Procedure

To configure the location of the Transaction Collector manually:

From the *dotnet.install.dir* directory, run the following command:

```
configdc ttserver -name [host name/ip address] -port [port number]
```

Note: When specifying an IPv6 IP address, ensure you enclose the address in square brackets, for example, `[::1]`.

What to do next

If configuration is complete, activate your configuration changes. See "Activating the configuration" for further information.

Enabling and disabling monitoring of ADO.NET interfaces

Customize which ADO.NET client interfaces you want to monitor.

About this task

By default, all supported ADO.NET interfaces are enabled for monitoring during installation of .NET Data Collector. To customize which interfaces are monitored, enable or disable monitoring for specific interfaces. If you disable monitoring of an interface, any associated application domain filter is retained for the next time that interface is enabled.

See Appendix I, "ADO.NET supported namespaces," on page 517 for a list of supported namespaces.

Procedure

To enable or disable monitoring for specific ADO.NET interfaces:

1. To enable monitoring, from the *dotnet.install.dir* directory, run the following command:

   ```
   configdc enableMonitor all | adsi | db2 | ldap | odbc | oledb | oracle | sql
   [-appdomain appdomain filter list]
   ```

2. To disable monitoring, from the *dotnet.install.dir* directory, run the following command:

   ```
   configdc disableMonitor all | adsi | db2 | ldap | odbc | oledb | oracle | sql
   ```

   See "Configuring .NET Data Collector using the configuration tool" on page 256 for further information about using this command.

What to do next

After configuring specific ADO.NET client interfaces to monitor, activate the configuration. See "Activating the configuration" for further information.

Activating the configuration

You must activate updated inactive configurations for the changes to take effect.

About this task

Remember: Reregister and reactivate the .NET Data Collector whenever you update the .Net framework.

Procedure

To activate an inactive configuration:

From the *dotnet.install.dir* directory, run the following command:

```
configdc activateconfig
```

What to do next

If Internet Information Services (IIS) server transactions are monitored and the Transaction Collector connection configuration was updated, restart the IIS server to complete the configuration activation.

If ASMX or WCF web services are monitored and the Transaction Collector connection configuration was updated, restart the process hosting the web service or reregister the .NET Data Collector using the **configdc unregister** and **configdc registerdc** commands.

Configuring .NET Data Collector using the configuration tool

You can use the command-line based configuration tool to configure .NET Data Collector.

Run all commands from the *dotnet.install.dir* directory.

Table 42. .NET Data Collector configuration commands

Command	Description
`configdc registerdc [-f]`	Register the .NET Data Collector. `-f` overrides any existing .NET profiler that is already registerd without any further prompts. If a profiler exists and the `-f` option is not specified, the .NET Tracking profiler is not configured. Run this command as an administrator.
`configdc unregisterdc`	Stop .NET Data Collector intercepting .NET transactions. Run this command as an administrator.
`configdc ttserver -name` *name* `-port` *port*	Validate and store the connectivity information for the Transaction Collector to which the data collector will be sending the tracking data. **Note:** When specifying an IPv6 IP address, ensure you enclose the address in square brackets, for example, `-name [::1]`.
`configdc logging [level ERROR | DEBUG | WARNING | INFO] [-size #] [-files #] [-tracing on|off]`	Enable or disable trace logging. Set tracing attributes such as the level of trace detail, the maximum trace log file size, and the maximum number of trace log files.

Table 42. .NET Data Collector configuration commands (continued)

Command	Description
`configdc enablemonitor [all | adsi | db2 | ldap | odbc | oledb | oracle | sql ] [-appdomain app domain list]`	Enable transaction monitoring for a particular ADO.NET interface. After installation, all interfaces are enabled for monitoring by default. Use `all` to enable all supported ADO.NET interfaces for transaction tracking. The `-appdomain` option restricts the specified ADO.NET interface monitoring to one or more specific .NET application domain names. If you are specifying more than one application domain name, separate each name with a semicolon (;). Use an asterisk (*) to enable all application domains; this is the value set at installation time. When an application domain list has been assigned to a particular ADO.NET interface, that value remains in the configuration until explicitly set to another value. If an interface is enabled for monitoring without specifying an application domain, the last application domain value specified for this interface is used.
`configdc disablemonitor [all | adsi | db2 | ldap | odbc | oledb | oracle | sql]`	Disable transaction monitoring for a particular ADO.NET interface. Use `all` to disable all supported ADO.NET interfaces for transaction tracking.
`configdc getconfig [-active]`	Display the properties file where you can add or remove .NET API calls for interception by .NET Data Collector. When you make changes to the configuration, you edit an inactive version of the properties file. When the changes are complete, activate the edited file using the `configdc activateconfig` command.
`configdc activateconfig`	Activate the updated configuration.

Configuring logging for .NET Data Collector

In ITCAM for Transactions V7.3, logging for .NET Data Collector is configured differently for each subcomponent of .NET Data Collector.

Performance of .NET Data Collector may be adversely affected when tracing is enabled. Disable tracing as soon as the required trace logs are gathered.

For the .NET profiler, configure logging using configdc. See “Configuring .NET Data Collector using the configuration tool” on page 256 for further information.

Internet Information Services ISAPI filter uses RAS1 logging. When the subcomponent is registered, the `ras1_iisdc.cfg` file is created in the *dotnet.install.dir*`\config` directory. Manually edit this file to change RAS1 logging behavior for the Internet Information Services ISAPI filter code and any other RAS1 based code running under either the `inetinfo` process or the `w3wp` process. Restart the IIS Service to implement the logging changes.

Note: Whenever you run **configdc registerdc**, it resets the ras1_iisdc.cfg file to its default values.

If .NET Data Collector is installed on a node that also has the ITCAM for SOA agent installed, ASMX/WCF trace and operation logging is controlled by the ITCAM for SOA agent. Use the ITCAM for SOA Take Action commands to adjust the logging behavior.

If only .NET Data Collector is installed on a node, ASMX/WCF trace and operation logging are read from dotNetDcConfig.properties.

To enable Transaction Tracking API trace logging for either the .NET Profiler or the ASMX/WCF SOAP handlers, set the following RAS1 environment variables. Set the environment variables locally, so that only the processes launched from the local command prompt use those environment variable values, or set the system environment variables globally. For example, set the following variables:

```
KBB_RAS1=DETAIL
KBB_RAS1_LOG="C:\IBM\ITM\TMAITM6\ktj\dotnet\logs\ktj.dotnet.isapi-.log" COUNT=03
 LIMIT=5 PRESERVE=1 MAXFILES=9
```

Note: If the ISAPI filter is registered, the RAS1 configuration in the ras1_iisdc.cfg file is used for code running under the inetinfo and w3wp processes, including the ASMX/WCF and .NET Profiler. If RAS1 tracing variables are globally set, these values supersede the values specified in the ras1_iisdc.cfg file.

Uninstalling .NET Data Collector

Uninstalling .NET Data Collector unregisters all three components, .NET profiler, ASMX/WCF, and IIS ISAPI filter.

Use the Windows **Add or Remove Programs** facility to uninstall .NET Data Collector from all computers on which it is installed.

You can also uninstall .NET Data Collector by running the following command:

```
dotnet.install.dir\bin\patchman.exe –m "dotnet.install.dir\.manifest.hive"
–r ".NET Data Collector 7.3"
```

Note: Ensure that you stop all monitored .NET applications and services before you uninstall .NET Data Collector.

Uninstalling if both .NET Tracking and .NET Data Collector are installed

In an environment with both .NET Tracking and .NET Data Collector installed, note the following:

- If both data collectors were installed to the same installation directory and one is uninstalled, you must reinstall the other data collector if it is still required.
- If the data collectors were installed in separate directories and one is uninstalled, reregister the ASMX/WCF SOAP handlers:
  - Reregister the ASMX/WCF handlers for .NET Tracking using the configNETDC.bat command.
  - Reregister the ASMX/WCF handlers for the .NET Data Collector using the configdc registerdc command. See “Configuring registration” on page 254 for further information.

Uninstalling if both ITCAM for SOA .NET data collector and .NET Data Collector are installed

In an environment with both ITCAM for SOA .NET data collector and .NET Data Collector installed, note the following:

- If you uninstall either data collector, the remaining data collector is disabled.
- If the ITCAM for SOA .NET data collector is uninstalled and .NET Data Collector remains, update the configuration:
  1. Edit the *dotnet.install.dir*\config\dotNetDcConfig.properties.inactive file and set the following property:

     ```
     default.kd4.enabled=false
     ```

  2. Run the `configdc activateconfig` command to activate the configuration change.
- Reregister the ASMX/WCF SOAP handlers:
  - Reregister the ASMX/WCF handlers for .NET Tracking using the `configNETDC.bat` command.
  - Reregister the ASMX/WCF handlers for the .NET Data Collector using the `configdc registerdc` command. See "Configuring registration" on page 254 for further information.

Preparing Tuxedo transaction tracking

To prepare Tuxedo transaction tracking, install and configure Tuxedo Tracking on each computer running Tuxedo whose applications you want to monitor.

Tuxedo Tracking is available in ITCAM for Transactions V7.1.0.2 and later.

Table 43 lists the inter-domain tracking supported by Tuxedo Tracking.

Table 43. Tuxedo Tracking - domain interactions supported

Domain interactions tracked	Notes
Tuxedo to and from Tuxedo	
Tuxedo to and from WebSphere MQ	Ensure that both Tuxedo Tracking and MQ Tracking are installed on the Tuxedo server.

Preparing your environment

To prepare your environment to monitor Tuxedo applications:

1. Identify all computers running applications on Tuxedo that you want to monitor.
2. On each of these computers, install Tuxedo Tracking.
3. When installation is complete, configure Tuxedo Tracking.
4. After configuration, enable Tuxedo Tracking where applicable.
5. Restart each Tuxedo server.

Prerequisites

Before installing Tuxedo Tracking, ensure that the following applications are installed and communicating:

- IBM Tivoli Composite Application Manager for Transactions V7.1.0.2 or later
- Tuxedo 9.0 or later

Supported operating systems

For information about supported operating systems:

1. Link to the required ITCAM for Transactions version from ITCAM for Transactions on Documentation Central
2. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**.
3. In the System requirements and prerequisites page, select Transaction Tracking from the Compatible software list.

Installing Tuxedo Tracking

Install Tuxedo Tracking on all computers running Tuxedo whose client and service transactions you want to track.

Before you begin

Tuxedo Tracking is included in the Transaction Tracking packages and is installed using a separate installer. The installer is named `tuxedo-ttmonitoring-`*`operating system-vbuildnumber`*.

For information about obtaining the latest version, see the Download information on the ITCAM for Transactions Information Center.

Ensure that you have read and understood the prerequisite information described in "Preparing Tuxedo transaction tracking" on page 259.

About this task

The following procedure describes the installation of Tuxedo Tracking on Windows systems. If you are installing on UNIX systems, the installation steps are similar but without a graphical user interface.

Procedure

To install Tuxedo Tracking on Windows systems:

1. Extract the Tuxedo Tracking installer, `tuxedo-ttmonitoring-`*`operating system-vbuildnumber`*, from the Transaction Tracking DVD.
2. Stop all Tuxedo applications and services, including `tlisten` and `tuxipc`.
3. Run the Tuxedo Tracking installer as an administrator.
4. On the **Welcome** window, click **Next**.
5. Enter the path to the directory where the Tuxedo Tracking files will be installed. For example, `C:\IBM\ITM\TMAITM6\ktj\tuxedo`. This directory is referred to as *tux.install.dir* in this documentation.
6. Click **Next**.
7. Enter the path to the Tuxedo installation directory and click **Next**. For example, `C:\bea\tuxedo10.0_vs2005`.
8. Enter the location of the Transaction Collector and click **Next**. For example, for IPv4 `tcp:collector ip address:5455`, or for IPv6 `tcp:[collector ip address]:5455`.
9. If tracking transactions between MQ and Tuxedo, enter the MQ Tracking shared library path that contains the file `TTDCCallback.dll` on Windows, or

`libTTDCCallback.{so,a}` on UNIX systems, and click **Next**. For example, `C:\IBM\ITM\TMAITM6\kth\lib\buildnumber`.

10. In the dialog box, select whether you want the installer to restart the computer after installation. The default is **No**.

    Note: You must restart the computer to complete the installation.

11. Click **Next** to begin the installation.

What to do next

Configure Tuxedo Tracking.

Configuring Tuxedo Tracking

Configure Tuxedo properties for Tuxedo Tracking.

Before you begin

Ensure that Tuxedo Tracking is installed.

About this task

The location of the Transaction Collector should have been set in the configuration file automatically during installation. If you were not prompted for the location of the Transaction Collector during installation of Tuxedo Tracking, you may need to do this manually.

Procedure

To configure Tuxedo Tracking manually:

1. Open `tux.install.dir/config/ttdcproxy.cfg` and ensure that the following entry is correct: `TTServerString=tcp:`*`collector address`*`:5455` .
2. On UNIX systems, start Tuxedo Tracking using the following command: `tux.install.dir/tuxdc.sh`.
3. On Windows systems, Tuxedo Tracking is installed as a service and starts automatically when you restart the computer.

What to do next

Enable Tuxedo Tracking.

Enabling Tuxedo Tracking

You must enable Tuxedo Tracking for each application whose transactions you want to track.

Before enabling Tuxedo Tracking, ensure that it is installed and configured.

Enabling Tuxedo Tracking

To enable Tuxedo Tracking:

- On Windows systems, restart the computer after installing Tuxedo Tracking
- On UNIX systems, re-source `tux.env` from Tuxedo home

Enabling Tuxedo Tracking on Windows in Tuxedo MP mode

To enable Tuxedo Tracking on Windows in Tuxedo MP mode if the Windows server is not the master, specify the following system environment variables:

- `TUXTT_CONFIG=`*`path to TT Tuxedo config file`*
- `TMTRACE=atmi:utrace`

To specify these system environment variables using the graphical user interface:

1. On the desktop, right-click **My Computer** and select **Properties**.
2. On the **Advanced** tab, click the **Environment Variables** button.
3. In the **System Variables** pane, click **New**.
4. Enter the variable name and value, as above, in the required fields and click **OK**.
5. Repeat steps 3 and 4 for the other variable.
6. Restart the computer. If you do not restart, the new variables will not be set.

Enabling transaction tracking through MQ

If your Tuxedo resource manager is using transactions and you want to track through MQ:

1. Ensure that MQ Tracking is installed on the Tuxedo server.
2. Set the **`QmgrSyncpointFilter`** parameter in the MQ API exits configuration file, `ttdcmqexits.cfg`, to the queue manager name. For example, **`QmgrSyncpointFilter = QMA,QMB`**.
3. Restart the queue manager.

Setting the MQ adapter name

If the MQ adapter is named anything other than TMQUEUE_MQM, update the `tuxtt.cfg` configuration file with the name of the MQ adapter. In the following line, replace *TMQUEUE_MQM* with the MQ adapter name:

```
SkipExecs = TMQUEUE_MQM, TMQUEUE_MQM.EXE
```

Configuring MQ Tracking to integrate with Tuxedo Tracking

If you are using a client connection, Tuxedo Tracking can only integrate with MQ Tracking if native (non-Java) MQ client-connection channel receive exits are enabled to track transactions as they enter and leave the MQ domain via client-connection channels.

Note: MQ client-connection channel receive exits are not required for applications that connect to MQ in bindings mode.

Configure channel receive exits for all client-connection channels used by Tuxedo Tracking to connect to queue managers that process the interactions between business applications that you want to track.

To configure channel exits for non-Java clients using client channel definition table (CCDT) files:

1. Ensure that MQ Tracking is installed on the Tuxedo server. Unless you need to track a local queue manager, you do not need to configure or run the TTDC Proxy for MQ.

2. On the server running the queue manager, modify the definition of the required client-connection channel. Configure the **RECVEXIT** field to use the required module directory, module name, and function name; and the **RECVDATA** field to use the data field:
   - Module directory:
     - Linux and UNIX 32–bit: `/opt/IBM/ITM/arch/th/lib/exits`
     - Linux and UNIX 64–bit: `/opt/IBM/ITM/arch/th/lib/exits64`
     - Windows x86: `C:\IBM\ITM\TMAITM6\kth\exits`
     - Windows x64: `C:\IBM\ITM\TMAITM6\kth\exits64`
   - Module name (found in the module directory):
     - Solaris: `libTTDCMqExitsClient_version`
     - Linux and other UNIX systems: `TTDCMqExitsClient_version_r`
     - Windows: `TTDCMqExitsClient_version`

       Note: On Windows, do not specify the `.dll` extension. On Linux and other UNIX systems, you can remove `_r` from the module name if your client is single threaded and built with the non-reentrant MQ client libraries.
   - Function name: `TTDCMqClientChannelRecvExit`
   - Data: the full path of the directory containing the `ttdcmqexits.cfg` file:
     - Linux and UNIX: `/opt/IBM/ITM/arch/th/config`
     - Windows: `C:\IBM\ITM\TMAITM6\kth\config`
3. Ensure that the CCDT file is accessible from the client, and that the following environment variables are set to indicate the location of the CCDT file:
   - `MQCHLLIB` - directory containing the CCDT file
   - `MQCHLTAB` - name of CCDT file, for example, `amqclchl.tab`

   Typically, the CCDT file is located on the server at `/var/mqm/qmgrs/qm_name/@ipcc` on Linux and UNIX systems, or at `C:\Program Files\IBM\WebSphere MQ\qmgrs\qm_name\@ipcc` on Windows systems.

For complete information about configuring channel exits, see the *WebSphere MQ System Administration Guide*.

Enabling Tuxedo Tracking manually

If you are not using `tux.env` on UNIX systems, or have your own environment, you can enable Tuxedo Tracking manually. Do the following for each Tuxedo application whose applications you want to track:

1. Shut down the application. For example, run `tmshutdown -y`.
2. Export `TMTRACE=atmi:utrace`.
3. Export `TUXTT_CONFIG=`*`tux.install.dir`*`/config/tuxtt.cfg`.
4. On Windows, set *`CANDLE_HOME`*`=c:\ibm\itm` to enable logging.
5. Restart the application. For example, run `tmboot -y`.
6. Run Tuxedo Tracking.

Tip: If you are enabling Tuxedo Tracking manually, add steps 2 and 3 to that `setenv` file so that tracking for that application is always enabled. Note that the customized `setenv` file should not be used by `ENVFILE` from `UBBCONFIG`. Using this file results in missing `tpinit()` and `tpsvrinit()` interceptions, which are crucial for tracking.

Uninstalling Tuxedo Tracking

After uninstalling Tuxedo Tracking, remove the custom environment variables manually.

To remove the custom environment variables on Windows systems:

1. Revert `TMTRACE` to the value it had before installing Tuxedo Tracking.
2. Remove `TUXTT_CONFIG`.
3. Restart the computer.

On UNIX systems, `TMTRACE` and `TUXTT_CONFIG` are located in the `tux.env` file in the Tuxedo installation directory. For example, `/opt/bea/tuxedo/10.0`. To remove the custom environment variables on UNIX systems:

1. Revert `TMTRACE` to previous value by removing `atmi:utrace` from the value.
2. Remove `TUXTT_CONFIG` from the file.
3. Restart Tuxedo.

Displaying ITCAM for SOA information in Transaction Tracking

You can use the ITCAM for SOA Log File Service to gather monitoring information collected in IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA) log files and convert it to display in Transaction Tracking workspaces and views.

The ITCAM for SOA integration is available in ITCAM for Transactions V7.1.0.2 and later.

Table 44 lists the intra-domain and inter-domain tracking supported by the ITCAM for SOA integration.

Table 44. ITCAM for SOA integration - domain interactions supported

Domain interactions tracked	Notes
SOA to and from SOA	ITCAM for SOA Log File Service displays topologies and other statistics in the Transaction Reporter workspaces.
SOA to WebSphere Message Broker	If WebSphere Message Broker is monitored by Data Collector for WebSphere Message Broker.
SOA to .NET Web Services	If .NET Web Services are monitored by .NET Data Collector.
SOA to WebSphere Application Server	If WebSphere Application Server is monitored by ITCAM for Application Diagnostics.

The ITCAM for SOA Log File Service, `ttdcproxy_soa`, takes data collected by the ITCAM for SOA log files and converts it to Transaction Tracking API events. The events are then sent to the Transaction Collector, and are available for display in the Transaction Tracking workspaces in the Tivoli Enterprise Portal.

Install `ttdcproxy_soa` on each computer with ITCAM for SOA installed whose information you want to display in Transaction Tracking workspaces.

Note: ITCAM for SOA V7.1.1 Fix Pack 1 and later provide native Transaction Tracking API data collectors for Message Broker and .NET. Data from Message

Broker and .NET data collectors is therefore no longer converted into corresponding Transaction Tracking API events by the ITCAM for SOA Log File Service. See the ITCAM for SOA Information Center for further information.

Installing ITCAM for SOA Log File Service

To install the ITCAM for SOA Log File Service:

1. Download the required package. See the Download information in the required ITCAM for Transactions Information Center on Documentation Central for further information. .
2. For each computer where ITCAM for SOA data collectors of interest are installed, sign in as a user with administrator privileges, and run the ITCAM for SOA Log File Service installer, `soa-ttmonitoring-`*`operating system`*`-v`*`buildnumber`*.
3. Enter the installation path for the ITCAM for SOA Log File Service.

   Note: The installer automatically detects the location of the ITCAM for SOA agent. If an ITCAM for SOA agent is not found, the installation continues and you will be prompted to manually set the **`LogFileMonitoringBaseDir`** property in `ttdcproxy.cfg`.
4. On Windows systems, restart the server.

To start the Transaction Tracking data collector proxy for ITCAM for SOA:

- On Windows systems, start and stop the ITCAM for SOA Log File Service service, `TTDC Proxy for ITCAM for SOA`, using the **Services** interface (**Start** > **Administrative Tools** > **Services**).
- On UNIX systems, run the command: **`#>/ ./ttsoa.sh start|stop`**

Note: Because both ITCAM for SOA WebSphere Message Broker data collector and Data Collector for WebSphere Message Broker provide user exits for monitoring WebSphere Message Broker, install only one of these components.

Sending ITCAM for SOA data to Transaction Tracking

After you have installed ITCAM for SOA Log File Service, on the same computer open the configuration file, `config/ttdcproxy.cfg`, in a text editor and add the name of the Transaction Collector to which the ITCAM for SOA Log File Service should send the converted Transaction Tracking API events. For example, `TTServerString = tcp:192.168.61.242:5455`.

Viewing ITCAM for SOA data in workspaces

ITCAM for SOA does not have a specific profile in the Application Management Configuration Editor. Instead, the **Other** profile is applied to the ITCAM for SOA tracking data by default.

The data gathered by ITCAM for SOA is displayed in Transaction Tracking workspaces in the Tivoli Enterprise Portal as follows:

- **Servers** workspaces display data about individual computers hosting ITCAM for SOA data collectors. The server name is the host name. For example, **soa-sol10z-wesb6200x64**.
- **Components** workspaces display data about the type of application server. For example, **WebSphere:APPLICATION_SERVER**.

- **Applications** workspaces display data about individual application servers. For example, **server1**.
- **Transactions** workspaces display data about individual ITCAM for SOA transactions, *Service Port Name - Service Port Namespace*. For example, **Customer - http://lc.retail.samples.wsm.ibm.com/**.

Uninstalling ITCAM for SOA Log File Service

To uninstall ITCAM for SOA Log File Service on Windows systems, use the **Add or Remove Programs** user interface.

To uninstall ITCAM for SOA Log File Service on UNIX systems, run the following command:

```
#>/proxy_install_dir/uninstall.sh
```

Note: After uninstalling the product some directories may remain, including `logs/` and `bin/`. If required, remove these directories manually after the uninstall process has finished.

Further information

For more information about installing and using ITCAM for SOA in the IBM Tivoli Monitoring environment, see the ITCAM for SOA Information Center.

Tracking WebSphere Application Server

You can track WebSphere Application Server transactions using one of two methods: ITCAM for Application Diagnostics; or using ARM (optionally by using the WebSphere Application Server Transaction Tracking (WASTT) data collector).

Consider the following information when deciding whether to use ARM or ITCAM for Application Diagnostics to track WebSphere Application Server:

- If you use ITCAM for Application Diagnostics, continue using the ITCAM for Application Diagnostics data collector for tracking.
- If your end-to-end composite application supports ARM, use ARM.

  For example, to track from Rational Performance Tester to IBM HTTP Server to WebSphere Application Server, use ARM in all components.

  Similarly, to track from IBM HTTP Server to WebSphere Application Server, use WASTT for WebSphere Application Server and ARM for IBM HTTP Server
- If your environment contains domains to track that do not support ARM, use ITCAM for Application Diagnostics.

  For example, use ITCAM for Application Diagnostics to track from:
  - IBM HTTP Server to WebSphere Application Server to MQ
  - IBM HTTP Server to WebSphere Application Server to CICS
  - IBM HTTP Server to WebSphere Application Server to IMS
- If you are using Byte Code Insertion based application management technology from another vendor on your WebSphere Application Server, use ARM. The ITCAM for Application Diagnostics data collector also uses Byte Code Insertion (BCI). Do not install multiple BCI technologies on the same Application Server.
- If you are enabling Transaction Tracking integration for the Web Response Time agent, use the ITCAM for Application Diagnostics data collector for tracking. See "Configuring Web Response Time" on page 164 for information about configuring the Web Response Time agent for data collection by selecting the

Enable Transaction Tracking Integration check box. See also the additional configuration steps required in ITCAM for Application Diagnostics to modify the `toolkit_custom.properties` file, as described in ITCAM for WebSphere Application Server: Data Collection and ITCAM for Transactions Integration Guide.

WebSphere Application Server settings

To enable the different types of tracking, ensure that the correct request metrics for WebSphere Application Server are set.

To enable the different types of tracking, set the request metrics for WebSphere Application Server as shown in Table 45. In the WebSphere Application Server administration console, use the navigation pane and select **Monitoring and Tuning** > **Request Metrics** to set the metrics.

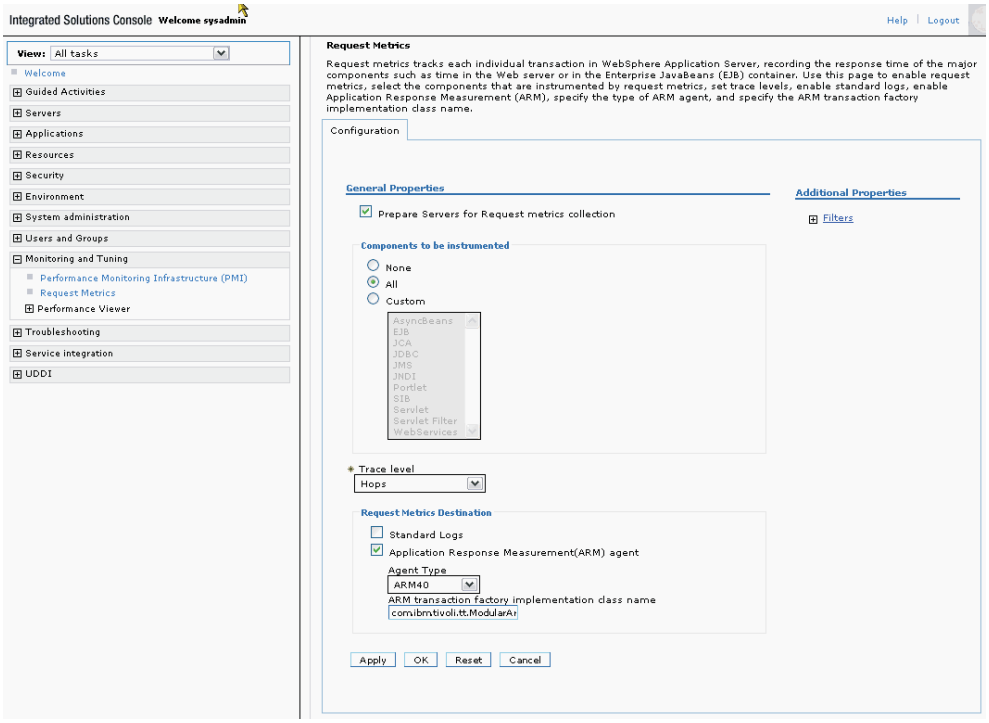


Table 45. WebSphere Application Server Request Metrics settings for different tracking methods

WebSphere Application Server setting	**Tracking WebSphere Application Server using ITCAM for Application Diagnostics**	**Tracking WebSphere Application Server using ARM**	**Tracking WebSphere Application Server using WASTT**
Prepare Servers for Request metrics collection	Not applicable	selected	selected
Components to be instrumented	None	All or custom	All or custom
Trace level	Not applicable	Hops, Performance_Debug, Debug	Hops, Performance_Debug, Debug
Request Metrics Destination	Standard Logs	**Application Response Measurement (ARM) agent**	**Application Response Measurement (ARM) agent**

Table 45. WebSphere Application Server Request Metrics settings for different tracking methods (continued)

WebSphere Application Server setting	Tracking WebSphere Application Server using ITCAM for Application Diagnostics	Tracking WebSphere Application Server using ARM	Tracking WebSphere Application Server using WASTT
Agent Type	Not applicable	**ARM40**	**ARM40**

ARM transaction factory implementation class name		`com.ibm.tivoli.tt.Arm TransactionFactory`
		
		Note: In versions of ITCAM for Transactions before V7.2, the following factory class was used:
		`com.ibm.tivoli.transperf.arm4. transaction.ARM40Transaction Factory`
		
		You can use either factory class.
			`com.ibm.tivoli.tt.Modular ArmTransactionFactory`
			
			This setting enables modules such as MQ exits, and JDBC tracking.

Note: WASTT sets the request metrics automatically. WASTT also requires other configuration, see "Configuring WASTT" on page 294 for further information.

Preparing ITCAM for Application Diagnostics

WebSphere Application Server can also be tracked by using ITCAM for Application Diagnostics. The ITCAM for Application Diagnostics data collector provides deep diagnostics.

The ITCAM for Application Diagnostics data collector can be configured to create and send Transaction Tracking API events to the Transaction Collector so that data is displayed in the ITCAM for Transactions workspaces.

Integrated Requests

Integration between ITCAM for Application Diagnostics Agent for WebSphere Applications data collector and ITCAM for Transactions supports all composite requests that generate events to Global Publishing Server (GPS).

It also supports top-level EJB and Custom Edge requests. In addition, it supports top-level Servlet and JSP requests to integrate with the ITCAM for Transactions Robotic Response Time agent (T6) and (WebSphere Application Server-supported) Web Servers with ARM-enabled plug-ins:

- CICS
- Custom Request
- EJB (including Message Driven Bean)
- IMS
- MQI, including MQ v7 JMS transactions
- RMI/IIOP
- Servlet/JSP
- Web Services

JDBC and JNDI nested requests are also supported.

WebSphere Portal Server is fully supported. Portlet transactions are displayed as single entities.

JMS links are displayed in the Topology View.

In the Topology View, an entity participating in transactions instrumented via both ITCAM for Application Diagnostics and ITCAM for Service Oriented Architecture will be displayed as a single node.

ITCAM for Application Diagnostics supports integration with IBM Optim Performance Manager. If this integration is enabled, the user can drill down from Transaction Tracking workspaces to the Optim Performance Manager extended monitoring console, to conduct end-to-end analysis of DB2 JDBC calls.

Enabling integration

To enable ITCAM Agent for WebSphere Applications data collector TTAPI integration, use the data collector configuration process.

Configure or reconfigure the ITCAM Agent for WebSphere Applications data collector for the application server instance, and select **Configure Transactions Integration**.

For details, see *IBM Tivoli Composite Application Manager Agent for WebSphere Applications Installation and Configuration Guide*.

For instructions on enabling ITCAM Agent for WebSphere Applications data collector and TTAPI integration on z/OS, see *IBM Tivoli Composite Application Manager Agent for WebSphere Applications Installation and Configuration Guide for z/OS*.

Disabling integration:

To disable ITCAM Agent for WebSphere Applications data collector and TTAPI integration, set the required property in the Toolkit.

To disable ITCAM Agent for WebSphere Applications data collector and TTAPI integration, set the following property in the `DCHOME/runtime/platform.node.server/custom/toolkit_custom.properties` file:

```
com.ibm.tivoli.itcam.dc.ttapi.enable=false
```

To disable integration of the ITCAM Agent for WebSphere Applications data collector with ITCAM for Transactions Web Response Time (T5) agent, set the following property in the `DCHOME/runtime/platform.node.server/custom/toolkit_custom.properties` file:

```
com.ibm.tivoli.itcam.dc.ttapi.wrm.servlet.enabled=false
```

After making these changes, restart the application server instance.

Monitoring JDBC and JNDI nested requests:

JDBC nested request monitoring is enabled by default for some monitoring levels. For other levels, you can enable JDBC and JNDI nested requests for the TTAPI.

JDBC nested request monitoring is enabled by default when the ITCAM Agent for WebSphere Applications data collector monitoring level, set in the Managing Server Visualization Engine, is L2 or L3. If the ITCAM Agent for WebSphere Applications data collector monitoring level is L1, you can enable the JDBC nested request feature in the ITCAM web console by following these steps:

1. Choose **Administration** -> **Server Management** -> **Data Collector Configuration** and select **Enable TTAPI for JDBC**.
2. In the **TTAPI JDBC DISABLED DATA COLLECTORS** panel, select the data collectors that you want to enable for JDBC nested requests.
3. Click **Apply**.

JNDI nested requests are monitored by default. To disable collecting JNDI information, set the following property in the `DCHOME/runtime/platform.node.server/custom/toolkit_custom.properties` file:

```
com.ibm.tivoli.itcam.dc.ttapi.jndi.enabled=false
```

After making these changes, restart the application server instance.

If exceptions (failed requests) occur within a reporting period, they are reported via TTAPI, and the status of the transaction is set to Fail. The user is able to inspect individual exceptions. To limit the amount of JDBC and JNDI exceptions displayed for a top-level transaction, set the following property in the `DCHOME/runtime/platform.node.server/custom/toolkit_custom.properties` file:

```
com.ibm.tivoli.itcam.dc.ttapi.maxExceptions=number
```

By default, this amount is limited to 10.

Enabling Optim Performance Manager integration:

If IBM Optim Performance Manager is installed, you can enable TTAPI integration between ITCAM for Application Diagnostics, Transaction Tracking, and Optim Performance Manager.

Optim Performance Manager provides detailed information about DB2 JDBC calls. If integration is enabled, you can "drill down" from transactions displayed in Transaction Tracking workspaces to the Optim Performance Manager console and dashboard to view deep database diagnostics information and detailed SQL statement performance data.

To enable Optim Performance Manager integration, set the following property in the `DCHOME/runtime/platform.node.server/custom/toolkit_custom.properties` file:

```
com.ibm.tivoli.itcam.dc.ttapi.jdbc.opm.enabled=true
```

If any monitored J2EE application changes the JDBC connection client attributes during an active session, also set the following property:

```
com.ibm.tivoli.itcam.dc.ttapi.jdbc.opm.clientinfo.reset=true
```

To make JDBC nested request information available when a data collector is running at MOD level 1, see “Monitoring JDBC and JNDI nested requests” on page 269.

When Optim Performance Manager integration is enabled, linkage of JDBC nodes to DB2 LUW nodes will be displayed in topology views, as displayed in Figure 10 on page 271.

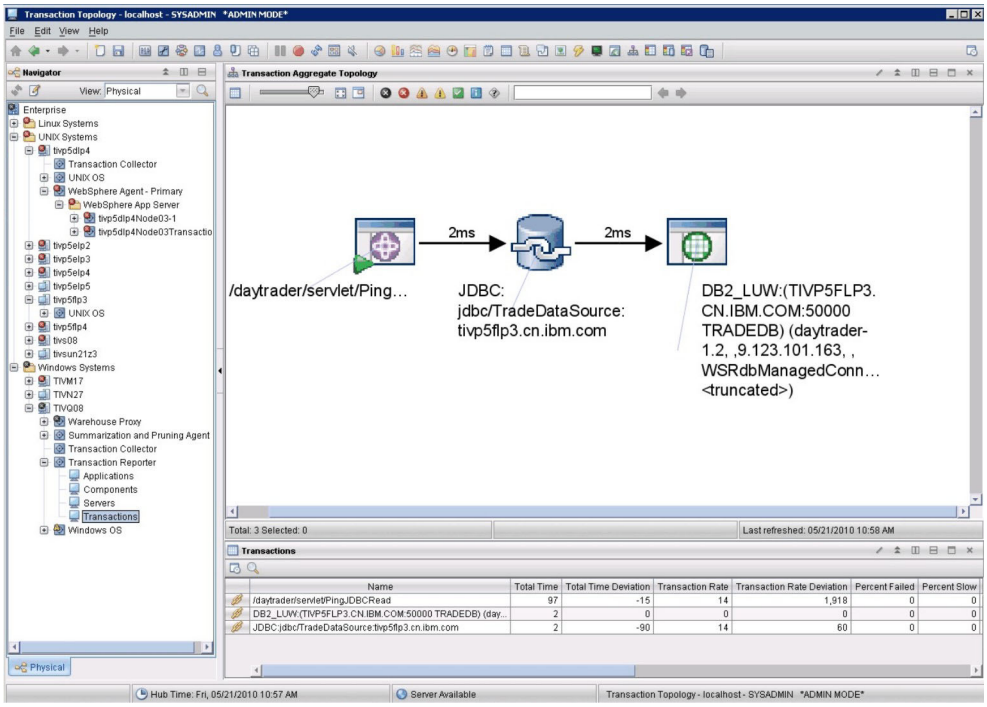


Figure 10. Example topology with Optim integration

To enable the **Database Diagnostics** workspace to display, complete the following steps:

1. Install the Optim Performance Manager plug-in for Tivoli Enterprise Portal, on the computer where your Tivoli Enterprise Portal Server is installed. This plug-in provides a workspace within Tivoli Enterprise Portal for using Optim Performance Manager to diagnose transaction performance problems.
2. Enable **Workspace Administration** for user system administration on your Tivoli Enterprise Portal. Make sure permissions **Workspace Administration Mode**and **Workspace Author Mode** are both selected.

Enabling and disabling MQ tracking:

You can use TTAPI to track MQ transactions (both MQI and JMS). To do this, you must enable MQ tracking using the Visualization Engine; it is disabled by default.

About this task

To track MQ requests on application server instances, make sure MQ tracking is enabled in the configuration applied to data collectors monitoring each of the instances. For each of the required data collector configurations, perform the following procedure.

Tip: For more information on configuring data collectors in the Visualization Engine, refer to *ITCAM for Application Diagnostics User Guide*.

Procedure

1. Log on to the Visualization Engine as a user with administrator permissions.
2. From the top navigation, click **Administration > Server Management > Data Collector Configuration**. The Configured Data Collector Overview page opens.

3. Click **Configuration Library** on the left navigation pane. The Data Collector Configuration List page opens.
4. Click the Modify icon next to the configuration you want to modify. The Modify page opens.

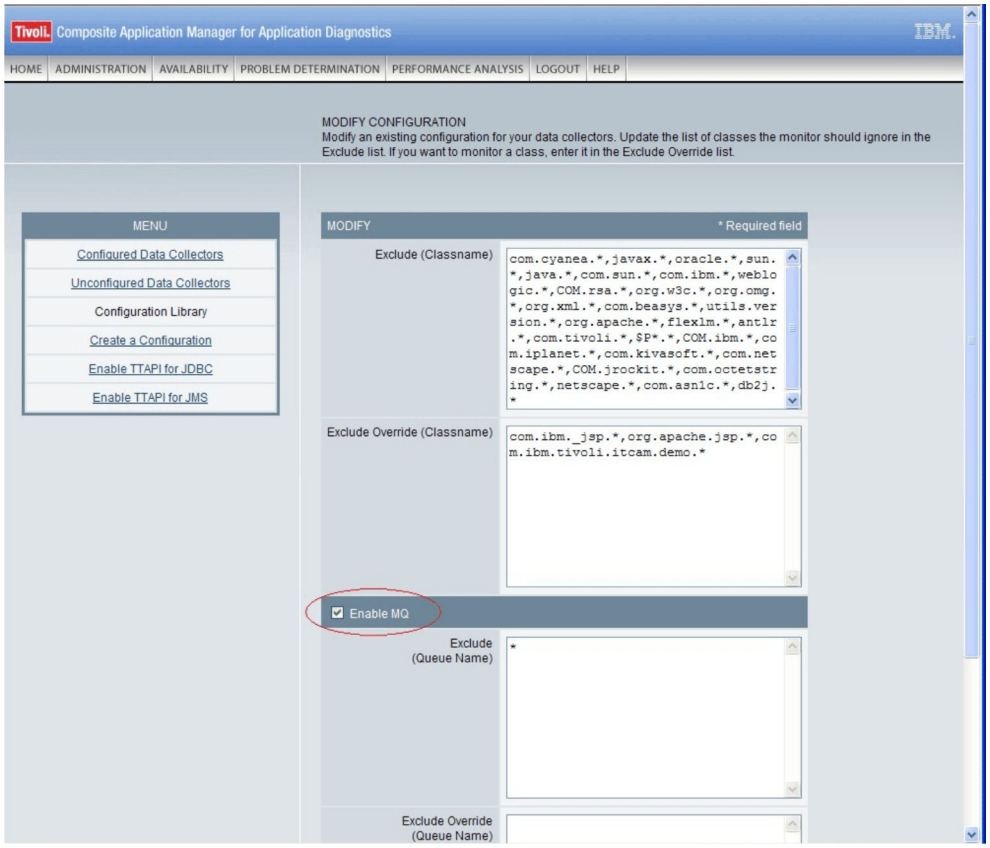


Figure 11. Enabling and disabling MQ tracking in the Modify Configuration window

5. You can perform any of the following changes:
   - To enable MQ transaction tracking for all queues, select the **Enable MQ** box and clear the **Exclude** window.
   - To disable MQ tracking for specific queues, enter the queue names in the **Exclude** window. You can use the * wildcard in the queue name; use the comma (**,**) to separate the queue names.
   - To enable MQ tracking for queues that would be disabled in the **Exclude** window, enter the queue names in the **Exclude Override** window.
   - To disable MQ transaction tracking for all queues, clear the **Enable MQ** box.
6. Click **Save** to save your modifications to the configuration. The Configured Data Collector Configuration List displays with the updated information.

Enabling and disabling JDBC tracking at MOD Level 1:

You can use TTAPI to track JDBC transactions. However, by default, an ITCAM Agent for WebSphere Applications data collector will not monitor JDBC transactions when it is set to MOD Level 1 by the Managing Server. If you need JDBC tracking on MOD L1, you must enable it.

About this task

Use the Visualization Engine to enable JDBC tracking at MOD L1.

Tip: For more information on configuring data collectors in the Visualization Engine, refer to *ITCAM for Application Diagnostics User Guide*.

Procedure

1. Log on to the Visualization Engine as a user with administrator permissions.
2. From the top navigation, click **Administration > Server Management > Data Collector Configuration**. The Configured Data Collector Overview page opens.
3. Click **Enable TTAPI for JDBC** on the left navigation pane. The Enable TTAPI for JDBC page opens.

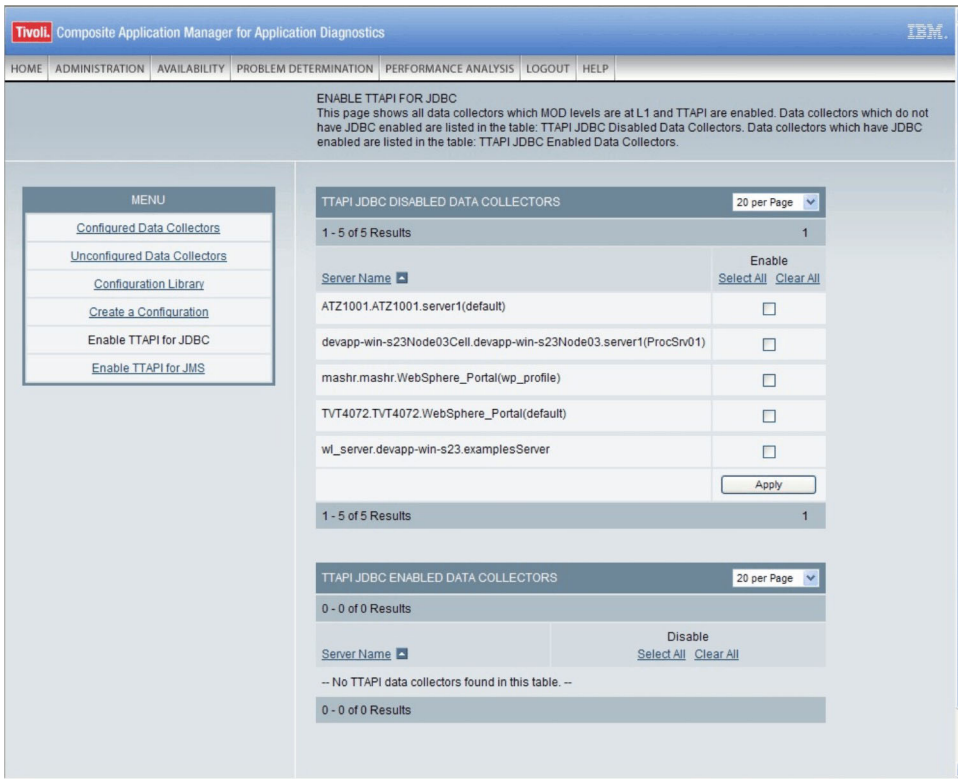


Figure 12. Enable TTAPI for JDBC window

4. You can perform any of the following changes:
   - To enable JDBC transaction tracking on MOD L1 for any monitored application servers, select the boxes next to the application server names in the **TTAPI JDBC DISABLED DATA COLLECTORS** table, and click the **Apply** button below this table.

- To disable JDBC transaction tracking on MOD L1 for any monitored application servers, select the boxes next to the application server names in the **TTAPI JDBC ENABLED DATA COLLECTORS** table, and click the **Apply** button below this table.

Enabling and disabling JMS tracking at MOD Level 1:

You can use TTAPI to display topology for JMS transactions. However, by default, an ITCAM Agent for WebSphere Applications data collector will not track JMS topology when it is set to MOD Level 1 by the Managing Server. If you need to view JMS topology on MOD L1, you must enable it.

About this task

Use the Visualization Engine to enable JMS tracking at MOD L1. (JMS tracking is enabled by default when the data collector is at MOD L2 or MOD L3).

Tip: For more information on configuring data collectors in the Visualization Engine, refer to *ITCAM for Application Diagnostics User Guide*.

Procedure

1. Log on to the Visualization Engine as a user with administrator permissions.
2. From the top navigation, click **Administration > Server Management > Data Collector Configuration**. The Configured Data Collector Overview page opens.
3. Click **Enable TTAPI for JMS** on the left navigation pane. The Enable TTAPI for JMS page opens.

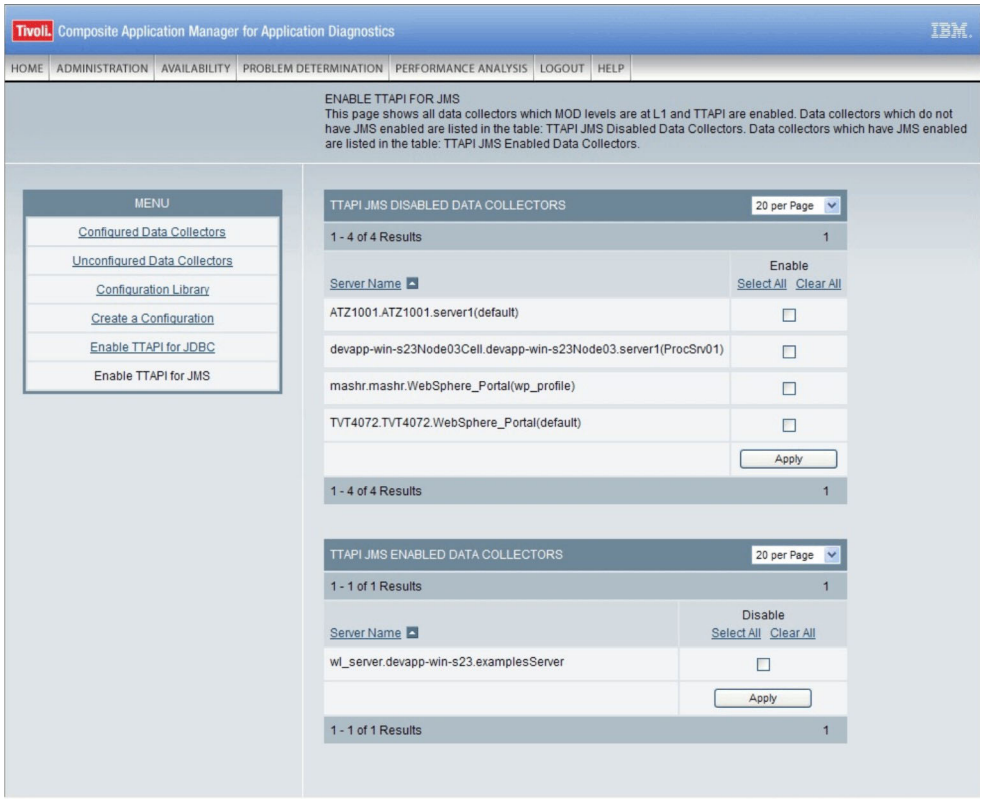


Figure 13. Enable TTAPI for JMS window

4. You can perform any of the following changes:
   - To enable JMS transaction tracking on MOD L1 for any monitored application servers, select the boxes next to the application server names in the **TTAPI JMS DISABLED DATA COLLECTORS** table, and click the **Apply** button below this table.
   - To disable JMS transaction tracking on MOD L1 for any monitored application servers, select the boxes next to the application server names in the **TTAPI JMS ENABLED DATA COLLECTORS** table, and click the **Apply** button below this table.

Displaying ITCAM for Application Diagnostics data in Transaction Tracking workspaces

After you have enabled integration between ITCAM for Application Diagnostics and Transaction Tracking, ensure that the ITCAM for Application Diagnostics data can be displayed in Transaction Tracking.

Servers, Components, and Applications workspaces:

Transaction Tracking Servers, Components, and Applications workspaces display aggregated transactions and request data.

Table 46 shows the values set by the ITCAM Agent for WebSphere Applications data collector so thatTransaction Tracking can identify the servers, components, and applications for its workspaces.

Table 46. ITCAM for Application Diagnostics name values for Transaction Tracking workspaces

Transaction Tracking workspace	**Value of Name**	**Notes**
Servers	Short DNS Name	
Components	WebSphere:Application_Server	
Applications	CellName.NodeName.ServerName (ProfileName)	For stand-alone servers, the default value set by the data collector
	ServerName(ProfileName)	For stand-alone application server instances, if `com.ibm.tivoli.itcam.dc.ttapi.` `appname.shortname=true` is specified in the server-specific `toolkit_custom.properties` file.
	CellName.NodeName.ServerName (ProfileName)^ClusterName (ClusterType) ClusterType can be either **Static** or **Dynamic**	For Network Deployment or Extended Deployment, the default value set by the data collector

Table 46. ITCAM for Application Diagnostics name values for Transaction Tracking workspaces (continued)

Transaction Tracking workspace	Value of Name	Notes
	ServerName(ProfileName) ^ClusterName(ClusterType)	For Network Deployment or Extended Deployment, if `com.ibm.tivoli.itcam.dc.ttapi.` `appname.shortname=true` is specified in the server-specific `toolkit_custom.properties` file.

These values remain the same for all integrated requests. Figure 14, Figure 15, and Figure 16 on page 277 show the ITCAM for Application Diagnostics values displayed in the Transaction Tracking workspaces.

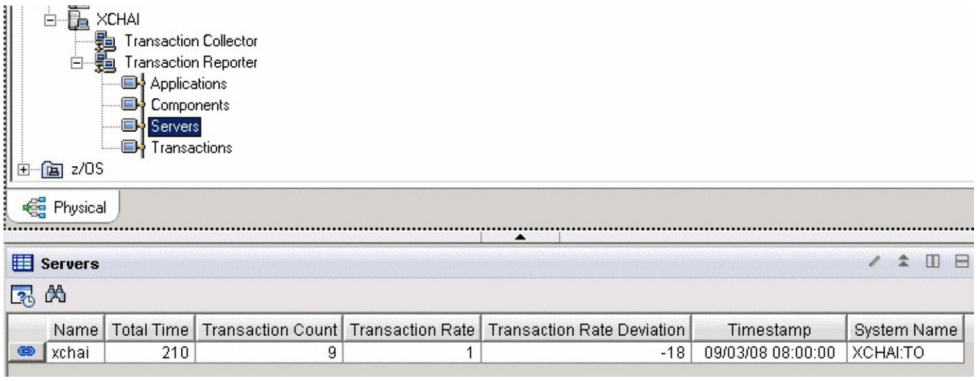


Figure 14. Servers workspace

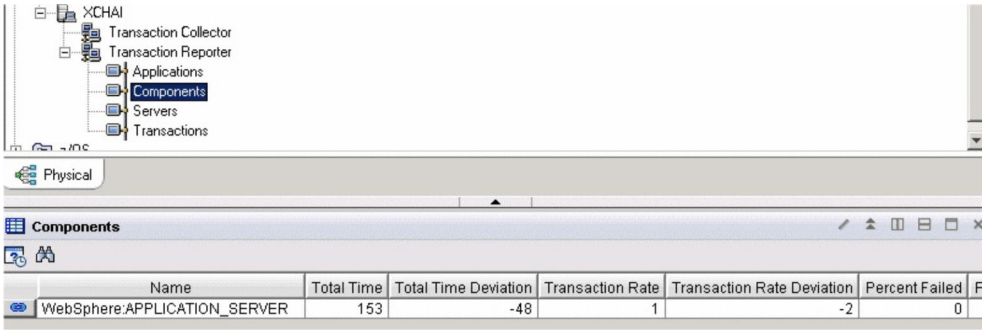


Figure 15. Components workspace

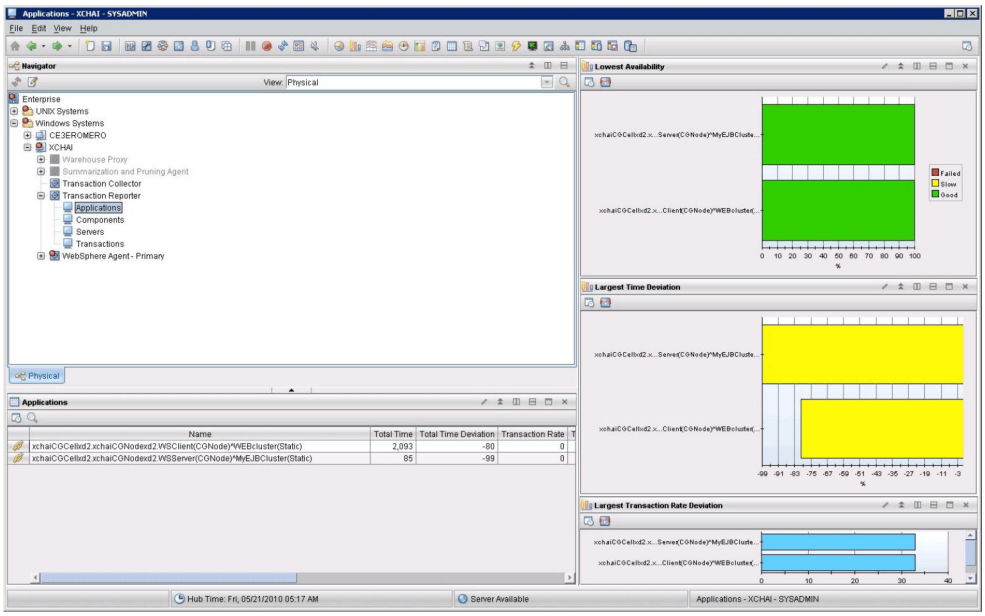


Figure 16. Applications workspace

Transactions workspace:

Use the Transactions workspace to view the aggregated transaction information, and to access additional information in ITCAM for Application Diagnostics from Transaction Tracking.

The Transactions workspace provides a list of transactions fitting certain criteria. Aggregated information is displayed for every transaction name.

To access the link menu for a transaction, right-click the chain icon at the left of the line in the transaction table. Use the link menu to view additional transaction information, including topology and response time statistics.

You can also use the link menu to access detailed information in ITCAM for Application Diagnostics:

- **Request Analysis** displays the Request Analysis workspace.
- **Diagnostics Recent Completed Requests** displays the recent request detail in the Managing Server Visualization Engine. This information is available only if the Deep Dive diagnostics infrastructure (Managing Server) is installed.

  Important: If the Managing Server used for monitoring an application server has been changed, the link to the Visualization Engine may not work. To enable the link again, perform the Forget Topology Take Action in the Transactions workspace. Then ensure that data is still sent from the data collector, and wait for four aggregation periods (by default, for 20 minutes). The aggregation period is set in the Translation Reporter configuration.

For more information about these workspaces, see *ITCAM for Application Diagnostics User Guide*.

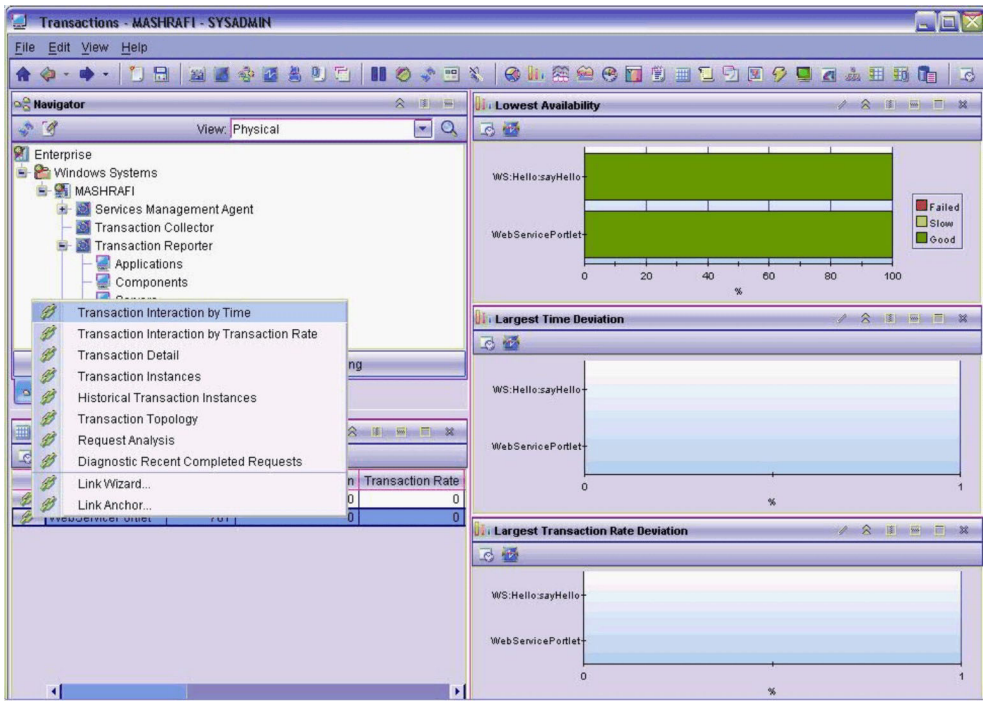


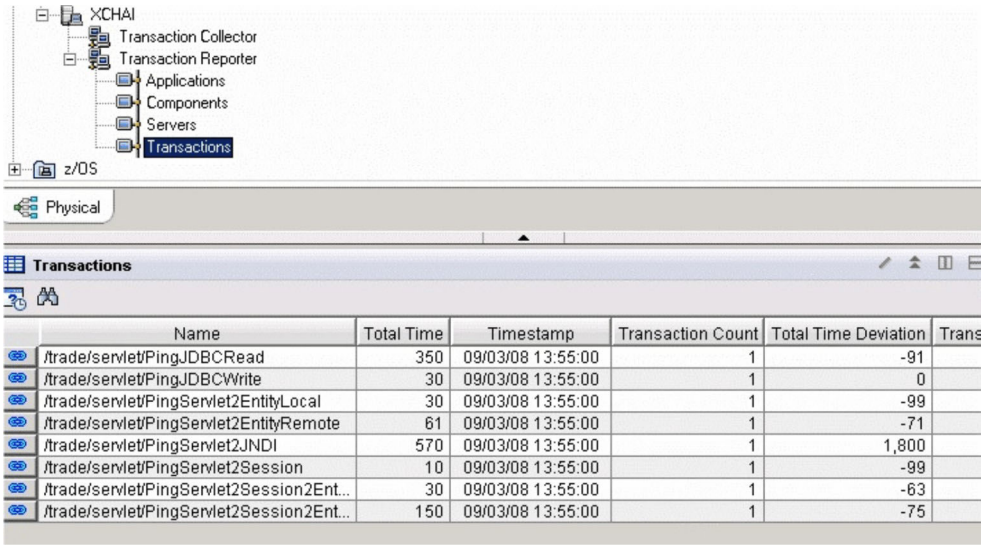
Figure 17. Transactions workspace

Servlet and JSP Request Integration:

Details on integrating the monitoring of Servlet and JSP requests.

Table 47. Values for Servlet and JSP requests

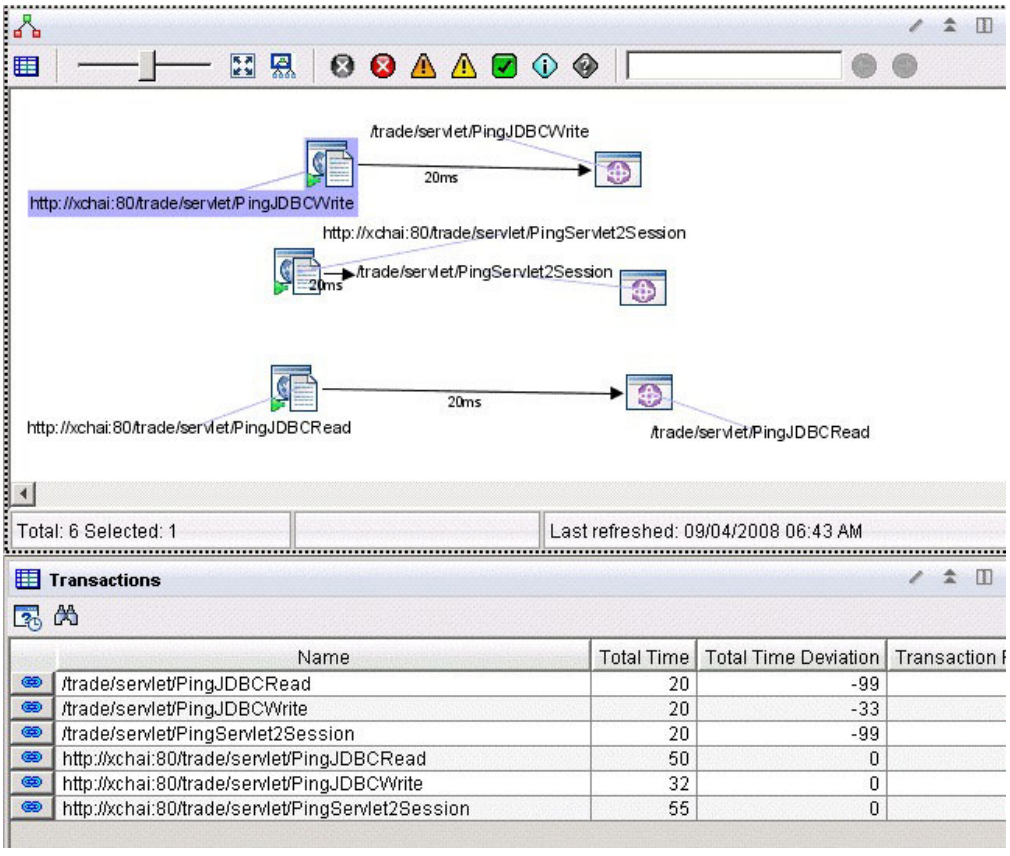
Transaction Tracking workspace	Value of Name	Notes
Transactions	URIT	The default value set by ITCAM Agent for WebSphere Applications data collector
	URI + QueryString	If `com.ibm.tivoli.itcam.dc.ttapi.servlet.include.querystring=true` is specifed in the server specific `toolkit_custom.properties` file



Name	Total Time	Timestamp	Transaction Count	Total Time Deviation	Trans
/trade/servlet/PingJDBCRead	350	09/03/08 13:55:00	1	-91	
/trade/servlet/PingJDBCWrite	30	09/03/08 13:55:00	1	0	
/trade/servlet/PingServlet2EntityLocal	30	09/03/08 13:55:00	1	-99	
/trade/servlet/PingServlet2EntityRemote	61	09/03/08 13:55:00	1	-71	
/trade/servlet/PingServlet2JNDI	570	09/03/08 13:55:00	1	1,800	
/trade/servlet/PingServlet2Session	10	09/03/08 13:55:00	1	-99	
/trade/servlet/PingServlet2Session2Ent...	30	09/03/08 13:55:00	1	-63	
/trade/servlet/PingServlet2Session2Ent...	150	09/03/08 13:55:00	1	-75	

Figure 18. Servlet and JSP transactions

Figure 19 shows the topology of the IBM HTTP Server (with ARM-enabled plugin) and WebSphere Application Server, displayed in the Transactions workspace.



Name	Total Time	Total Time Deviation	Transaction F
/trade/servlet/PingJDBCRead	20	-99	
/trade/servlet/PingJDBCWrite	20	-33	
/trade/servlet/PingServlet2Session	20	-99	
http://xchai:80/trade/servlet/PingJDBCRead	50	0	
http://xchai:80/trade/servlet/PingJDBCWrite	32	0	
http://xchai:80/trade/servlet/PingServlet2Session	55	0	

Figure 19. Servlet and JSP topology in Transactions workspace

On Figure 20 on page 280, the topology view of the Components workspace shows interaction between the HTTP Server and WebSphere Application Server:

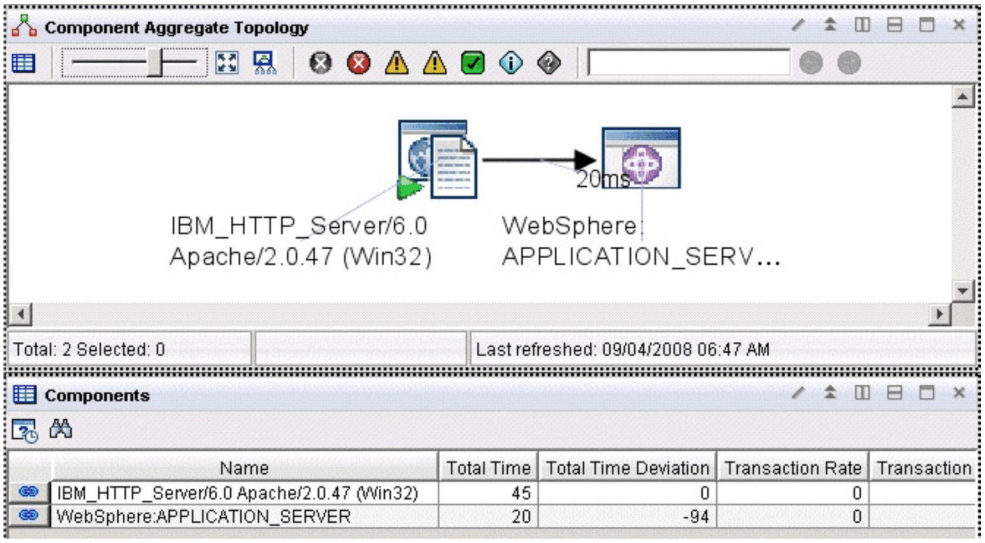


Figure 20. Servlet and JSP topology on Components workspace

On Figure 21, the topology view of the Applications workspace shows interaction between one instance of the HTTP Server and one instance of WebSphere Application Server:

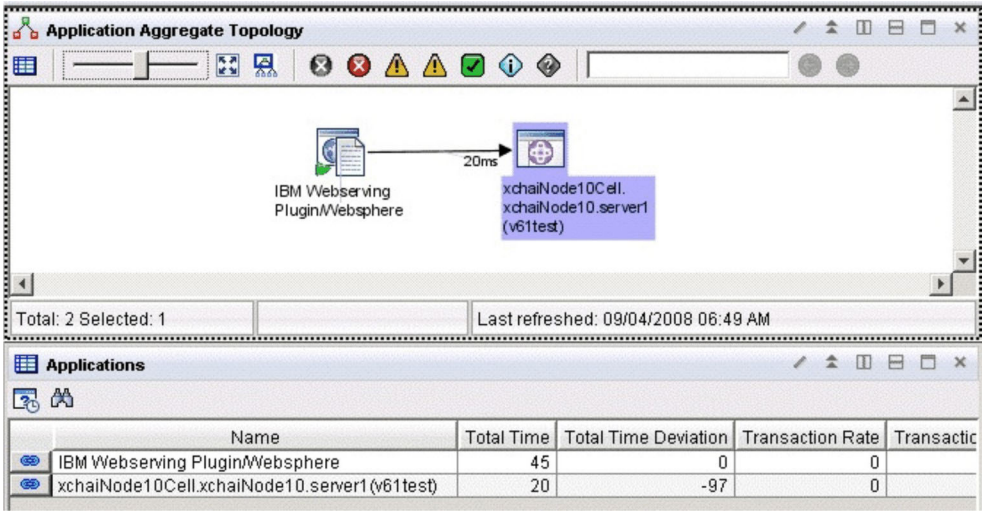


Figure 21. Servlet and JSP topology on Applications workspace

RMI and IIOP Request Integration:

Details on integrating the monitoring of RMI and IIOP requests.

Table 48. Values for RMI and IIOP requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	RMI Client: Invoking Servlet/JSP URI or ClientRequestInterceptor:MethodName	
	RMI Server: ServerRequestInterceptor:MethodName	

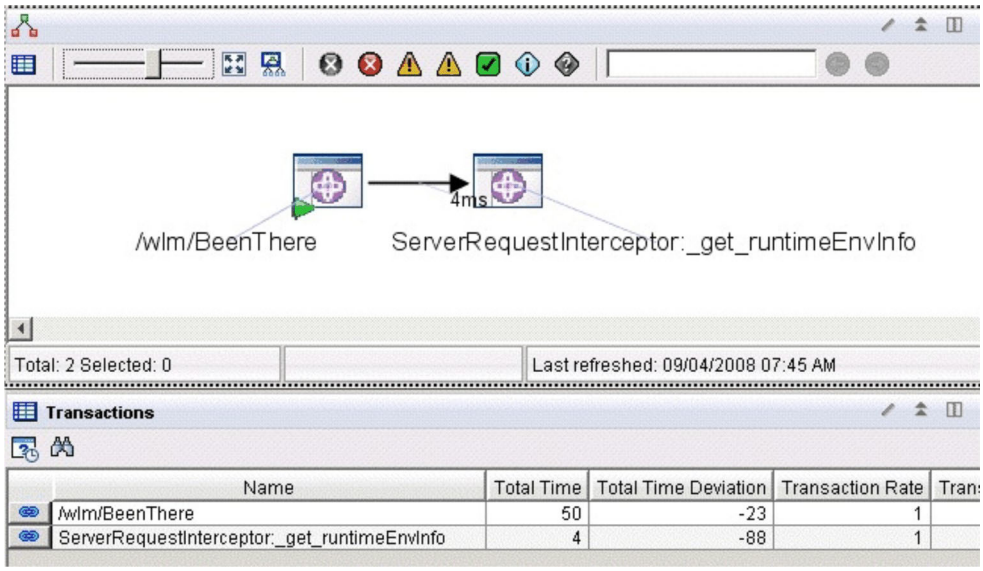


Figure 22. RMI/IIOP topology view in Transactions workspace

Web Services request integration:

Details on integrating the monitoring of Web Services requests.

Table 49. Values for Web Services requests

Transaction Tracking workspace	Value of Name Column	Note
Transactions	Client Side: Invoking Servlet/JPS or WS:WebServicePort:OperationName	
	Server Side: WS:WebServicePort:OperationName	

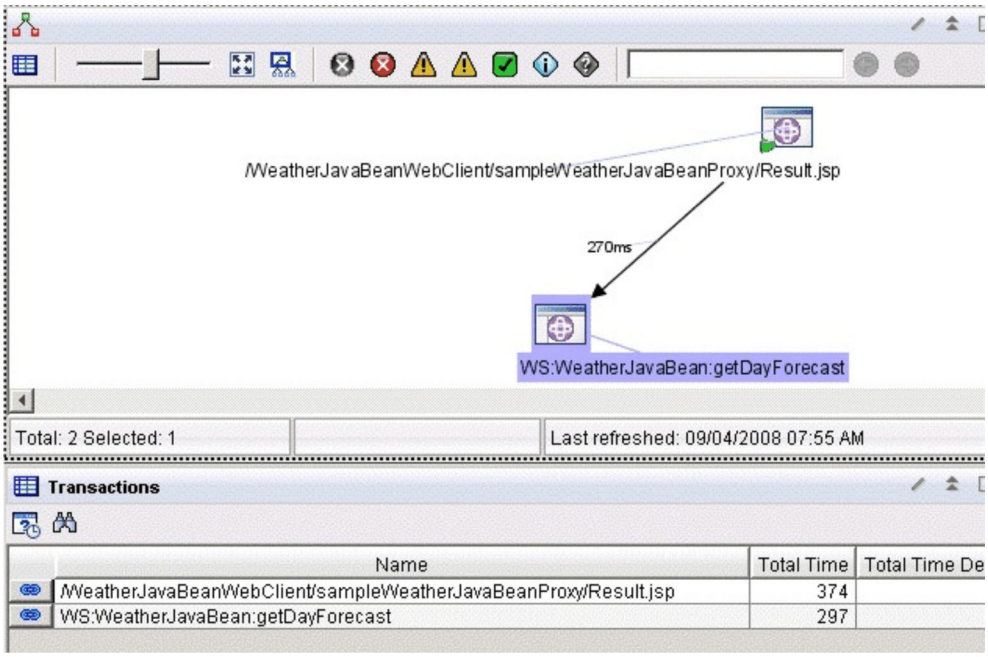


Figure 23. Web Services topology view in Transactions workspace

MQI request integration:

Details on integrating the monitoring of MQI and MQv7 JMS requests.

Table 50. Values for MQI and MQv7 JMS requests

Transaction Tracking workspace	**Value of Name**	**Notes**
Transactions	PUT/GET: Invoking Servlet/JPS or QueueManagerName:QueueName	

For MQI requests except MQ v7 JMS requests, details are available in the Managing Server Visualization Engine. For MQ v7 JMS requests, details are **not** available in the Visualization Engine.

To track MQI transactions, you must enable MQ tracking using the Visualization Engine. See “Enabling and disabling MQ tracking” on page 271.

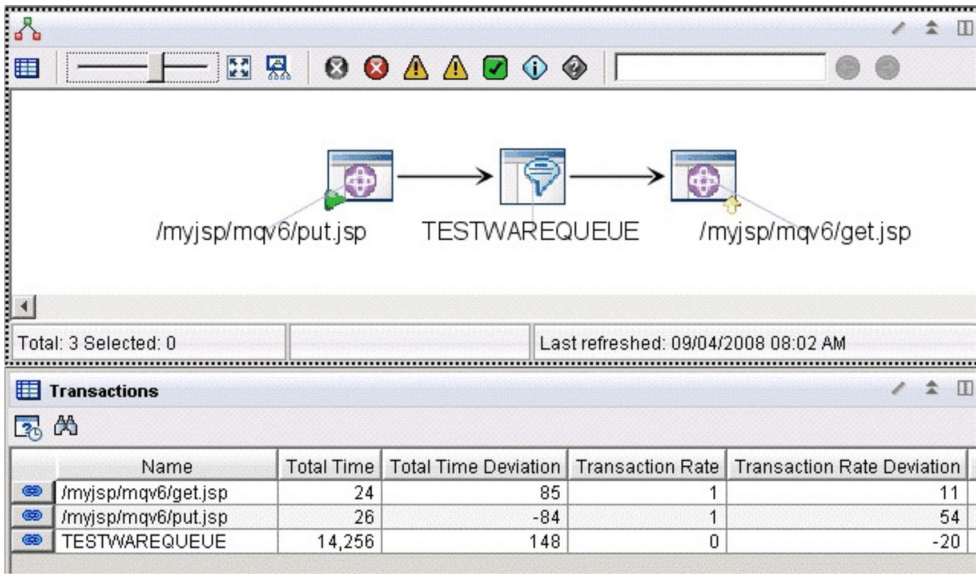


Figure 24. MQI topology view in Transactions workspace

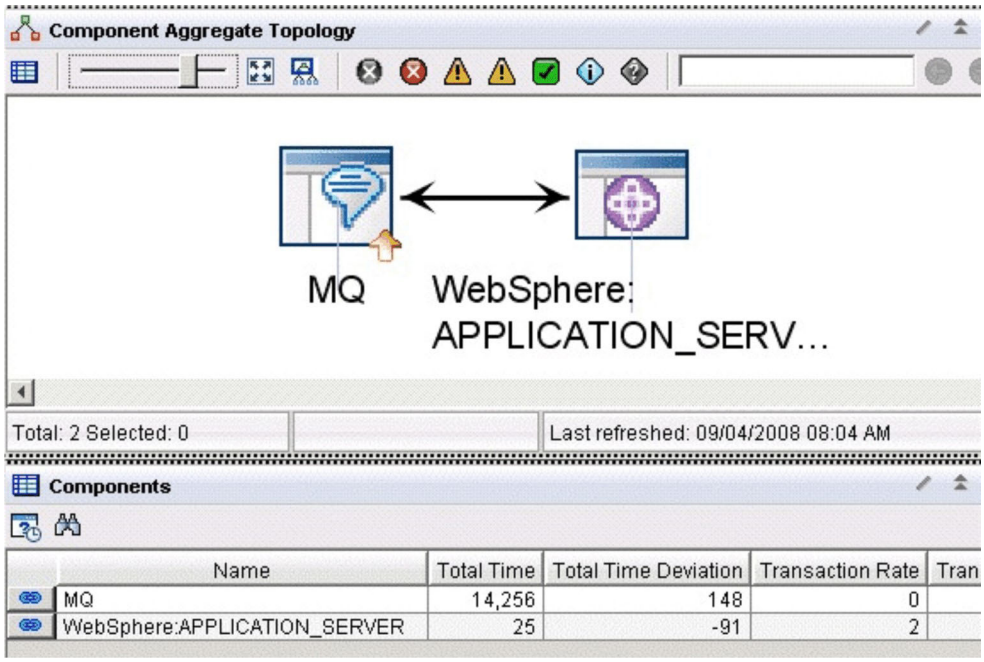


Figure 25. MQI topology view in Components workspace

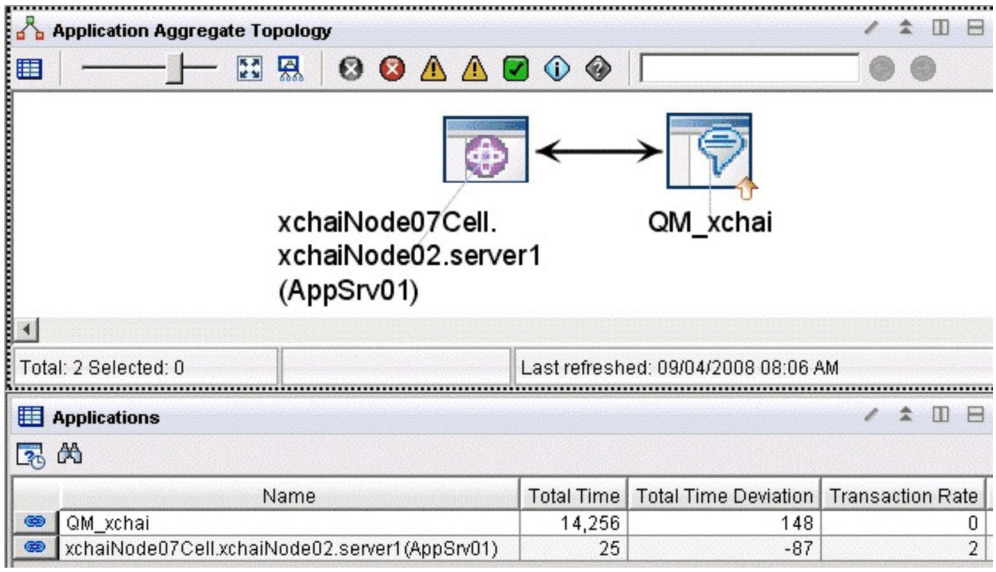


Figure 26. MQI topology view in Applications workspace

CICS integration:

Details on integrating the monitoring of CICS requests.

Table 51. Values for CICS requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	WebSphere client: The invoking request URI.	Represents processing by CICS components.
	CTG/CICS: "CSMI"	

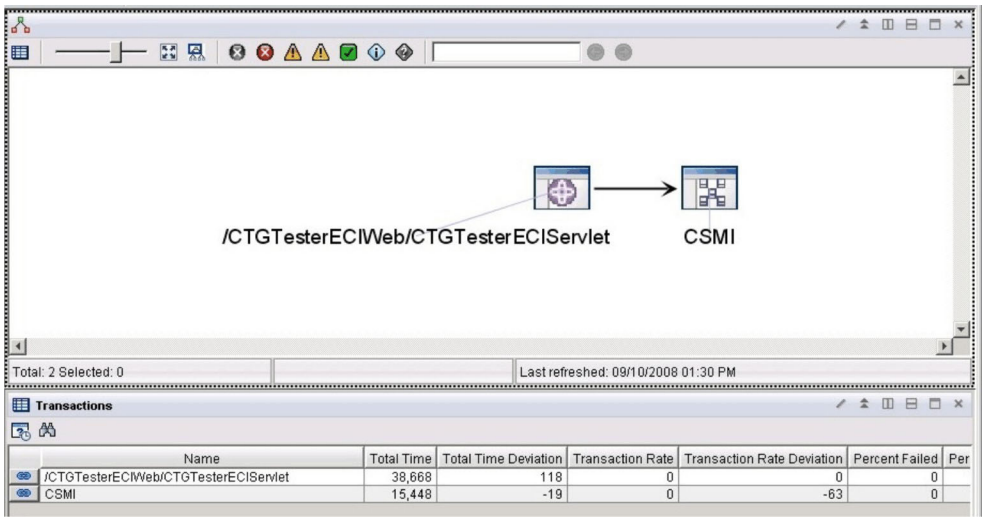


Figure 27. CICS topology view in Transactions workspace

IMS integration:

Details on integrating the monitoring of IMS requests.

Table 52. Values for IMS requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	WebSphere client: The invoking Servlet or JSP request URI, or IMS Connect component names.	

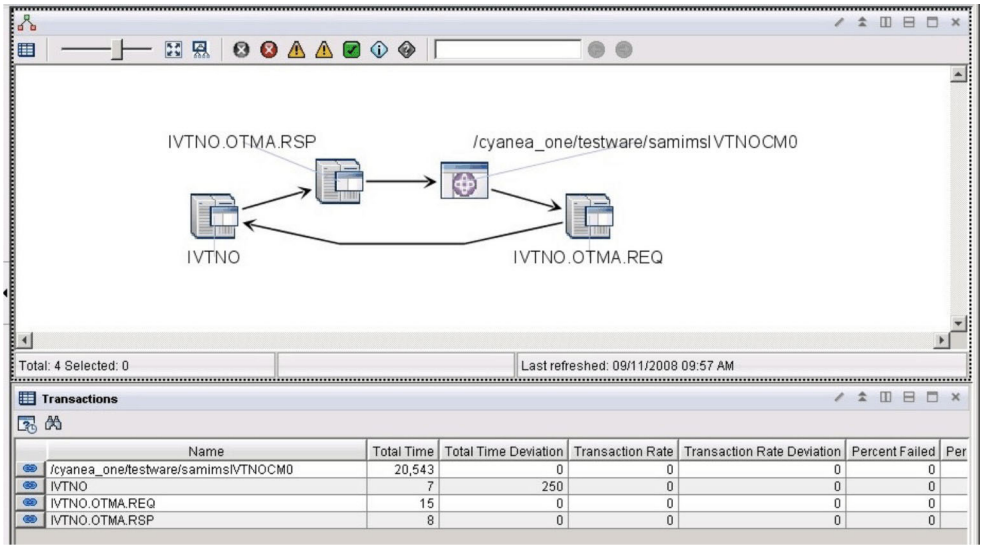


Figure 28. IMS topology view in Transactions workspace

EJB integration:

Details on integrating the monitoring of EJB requests.

Table 53. Values for EJB requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	EJB ClassName.methodName	Only the top-level EJB request is displayed.

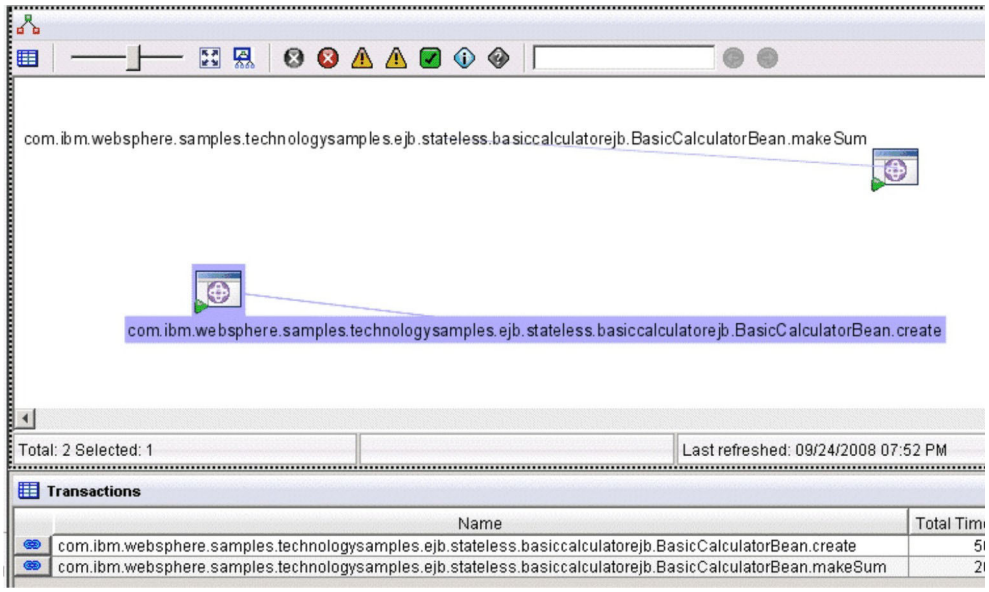


Figure 29. EJB topology view in Transactions workspace

Message Driven Bean integration:

Details on integrating the monitoring of Message Driven Bean requests.

Table 54. Values for Message Driven Bean requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	MDB className.onMessage	

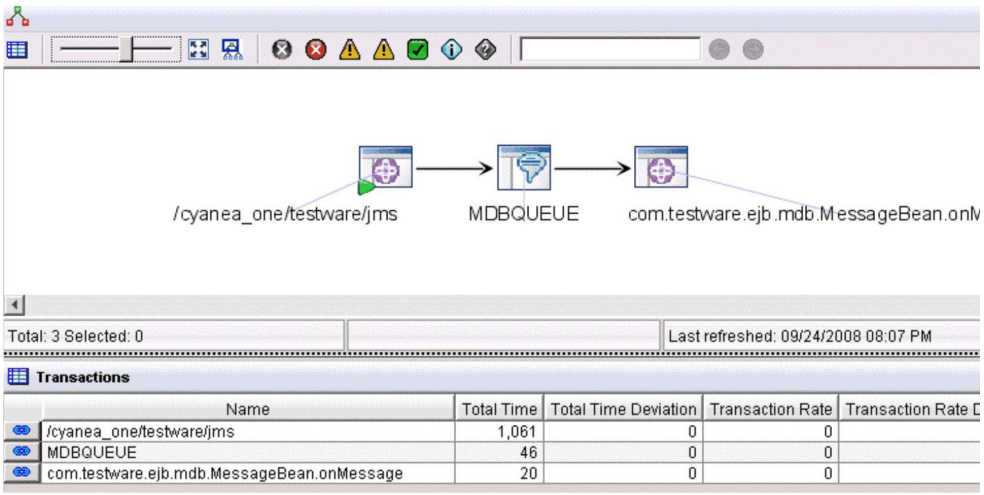


Figure 30. Message Driven Bean topology view in Transactions workspace

Custom request integration:

Details on integrating the monitoring of custom requests.

Table 55. Values for custom requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	RequestName defined in custom request configuration XML file.	Only the top-level Custom Request is displayed.

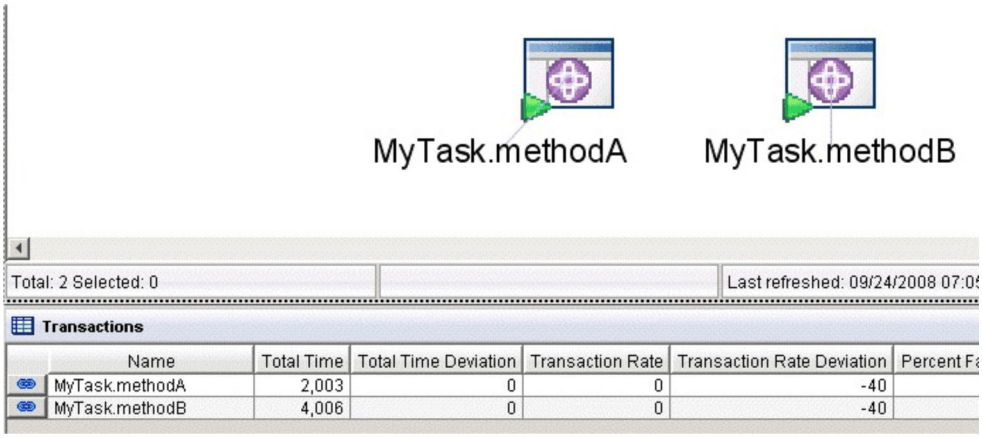


Figure 31. Custom request topology view in Transactions workspace

JDBC nested request integration:

Details on integrating the monitoring of JDBC nested requests.

Table 56. Values for JDBC nested requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	JDBC:dataSourceName JDBC:dataSourceName:hostName	

Attention: the host name is only reported if JDBC type 4 drivers are used.

To track JDBC transactions when a data collector is at MOD Level 1, you must enable this tracking in the Visualization Engine; see "Enabling and disabling JDBC tracking at MOD Level 1" on page 273. If you do not enable it, JDBC transactions will be tracked for an application server instance only when it is monitored at MOD L2 or MOD L3. The monitoring level can be set in the Visualization Engine; see *ITCAM for Application Diagnostics User Guide*.

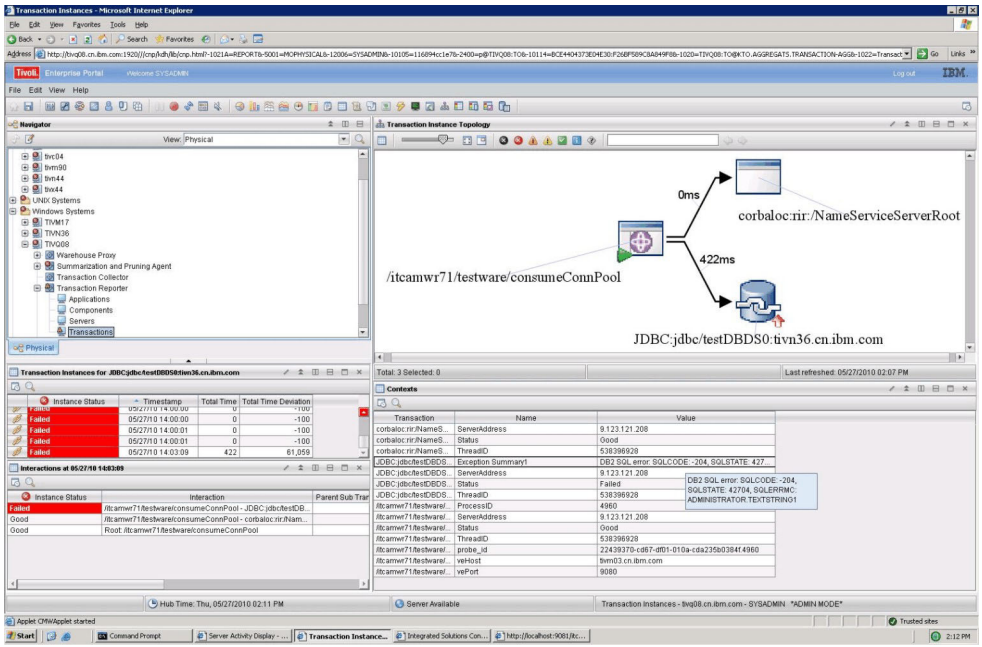


Figure 32. JDBC nested request view in Transactions workspace

Important: If a JDBC transaction reports any failures within a reporting period, the percentage of failed transactions will be displayed for the transaction. You can view individual successful and failed instances using the Transaction Instances view.

JNDI nested request integration:

Details on integrating the monitoring of JNDI nested requests.

Table 57. Values for JNDI nested requests

Transaction Tracking workspace	Value of Name	Notes
Transactions	JNDI:transactionName	

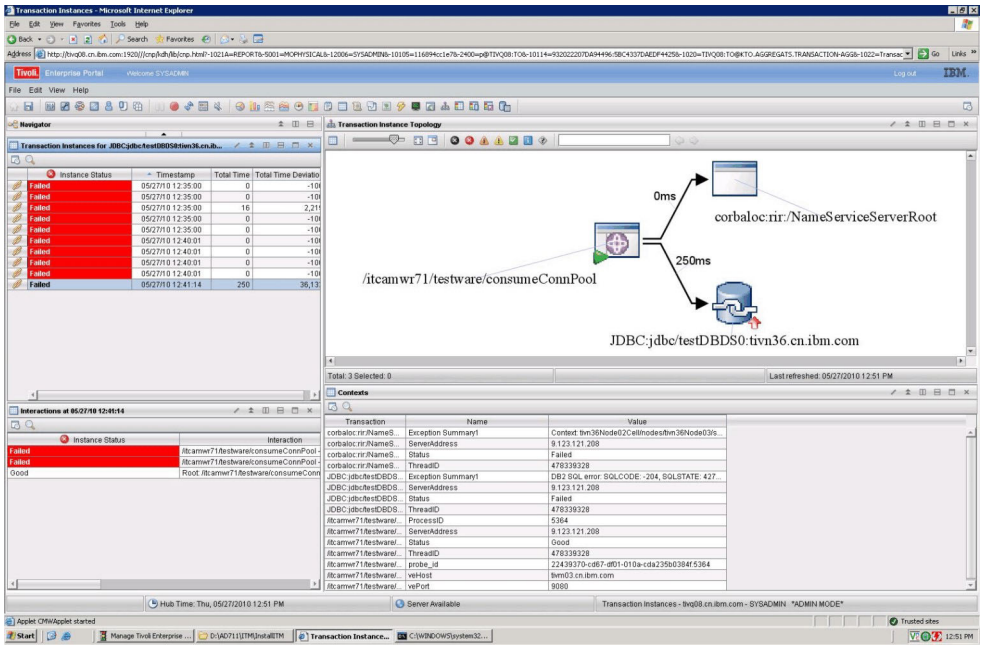


Figure 33. JNDI nested request view in Transactions workspace

Important: If a JNDI transaction reports any failures within a reporting period, the percentage of failed transactions will be displayed for the transaction. You can view individual successful and failed instances using the Transaction Instances view.

JMS messaging topology integration:

Details on support for JMS messaging topology.

When JMS tracking is enables (see "Enabling and disabling JMS tracking at MOD Level 1" on page 274), JMS links will be displayed between the following top-level request types:

- EJB, see "EJB integration" on page 285.
- Servlet
- Custom Request

The following JMS providers are supported:

- WebSphere SIBus
- WebSphere MQ
- WebLogic JMS server

JMS topology for WebSphere MQ transactions is not displayed to avoid duplication. If the transaction target is set to MQ, enable MQ tracking to display its topology (see "Enabling and disabling MQ tracking" on page 271). JMS topology will be displayed for queue transactions where the target is set to JMS (in WebSphere Application Server 7, where **appending RFH version 2 headers** is enabled for the queue).

The following JMS messaging scenarios are supported for TTAPI:

- **queue Sender and queue Receiver**. A top-level request invokes the queueSender API to send a message to a queue. A top-level request invokes the

queueReceiver API to receive a message from the queue. The application URLs displayed for the sender and receiver are the URLs of the top-level requests.

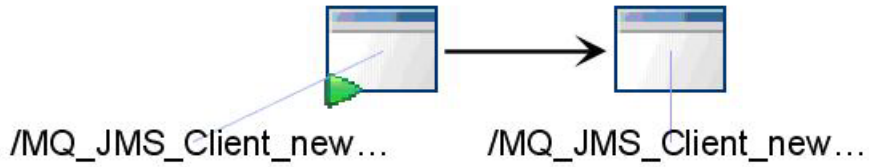


Figure 34. JMS example topology: queue sender and queue receiver

- **Topic Publisher and Topic Subscriber**. A top-level request invokes the TopicPublisher API to send a message to a Topic. One or several top-level requests may invoke the TopicSubscriber API to receive a message from the Topic. The application URLs displayed for the sender and receivers are the URLs of the top-level requests.

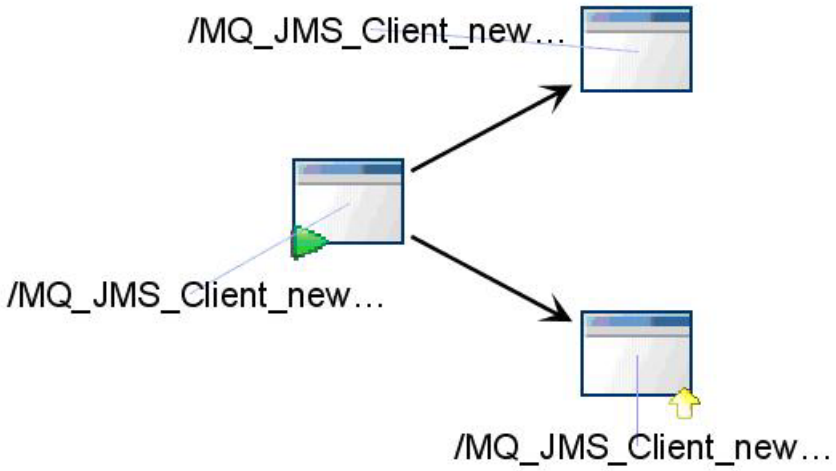


Figure 35. JMS example topology: topic publisher and topic subscriber

- **Message Sender and Message Driven Bean**. A top-level request sends a message to a queue or Topic. A Message Driven Bean that listens to the queue or Topic gets a callback on its onMessage method and receives the message. For the sender, the application URL is displayed; it is the URL of the top-level requests. For the receiver, the Message Driven Bean class name and method name are displayed.

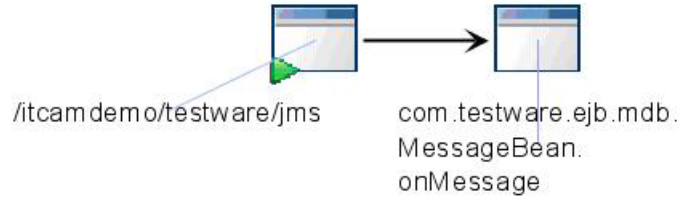


Figure 36. JMS example topology: message sender and message driven bean

Logging and tracing:

Details on logging and tracing for the ITCAM for Application Diagnostics data collector and Transaction Tracking Application Programming Interface (TTAPI) integration.

Logging and tracing for the data collector:

To enable logging and tracing for the data collector for the ITCAM for Application Diagnostics and TTAPI integration, update the toolkit files.

Add the following entries to the `DCHOME/toolkit/etc/cynlogging.properties` file to trace the data collector and Transaction Tracking Application Programming Interface integration:

```
# dc and ttapi integration tracing
CYN.trc.shared.datacollector.ttapi.TTAPIUtil.level=DEBUG_MAX
CYN.trc.shared.datacollector.ttapi.TTAPIUtil.logging=true
```

Add the following entry to the `DCHOME/runtime/<platform.node.server>/custom/toolkit_custom.properties` file to record event information when the writing of the events fail:

```
com.ibm.tivoli.itcam.dc.ttapi.logExceptionEventRecs=true
```

The standard logging locations are:

For Windows® systems:

```
C:\Program Files\IBM\tivoli\common\CYN
```

For Linux® and Unix® systems:

```
/var/ibm/tivoli/common/CYN
```

Logging and tracing for the Transaction Tracking API and Transaction Collector:

Transaction Tracking API can use the IBM Tivoli Monitoring standard RAS1 logging package to log error and debug messages at significant points when initializing, shutting down, and sending events to a Transaction Collector.

Logging can be controlled by the environment variables described in Table 58.

Table 58. Logging environment variables

Environment variable	Description
KBB_RAS1=ALL	Enable logging of all messages.
KBB_RAS1=ERROR	Enable logging of error messages
KBB_RAS1=	Disable all message logging. This is the default value.
KBB_RAS1_LOG=	Log to standard output.
KBB_RAS1_LOG=. . .	Set the log file name and other parameters. See the information following this table.
KBB_VARPREFIX=%	Set the prefix for variables specified in KBB_RAS1_LOG

KBB_RAS1_LOG uses the following format:

```
KBB_RAS1_LOG=filename [INVENTORY=inventory filename] [COUNT=count]
   [LIMIT=limit] [PRESERVE=preserve] [MAXFILES=maxfiles]
```

where:

count is the maximum number of log files to create in one invocation of the application.

inventory filename is a file in which to record the history of log files across invocations of the application.

limit is the maximum size per log file.

maxfiles is the maximum number of log files to create in any number of invocations of the application. This only takes effect when *inventory filename* is specified.

preserve is the number of log files to preserve when log files wrap over <*count*>.

For WebSphere Application Server, you can set the environment variables for RAS1 tracing through AdminConsole, as shown in the following example:

1. Navigate to **Server** > **Application Servers** and select the *ServerName*.
2. In the Configuration tab, navigate to **Server Infrastructure** > **Java and Process Management** > **Process Definition** > **Additional Properties: Custom Properties**.
3. Set the following environment variables:

```
KBB_RAS1=ALL
KBB_RAS1_LOG=c:\itcam71\tt71\ras1.l
```

If your server instance belongs to a Network Deployment cell, synchronize your change with NodeAgent.

Preparing WebSphere Application Server Transaction Tracking (WASTT)

To prepare WASTT, install and configure WASTT on each computer running WebSphere Application Server.

WASTT is available in ITCAM for Transactions V7.1.0.2 and later.

Table 59 lists the intra-domain and inter-domain tracking supported by WASTT.

Table 59. WASTT - domain interactions supported

Domain interactions tracked	Notes
WASTT to and from WASTT	
WASTT to WebSphere MQ	Client and Bindings mode are supported. Enable ARM in WebSphere Application Server.
ARM to WASTT	For example, IBM HTTP Server to WebSphere Application Server, using WASTT for WebSphere Application Server and ARM for IBM HTTP Server.

Preparing your environment

To prepare your environment to monitor WebSphere Application Server using WASTT:

1. Identify all computers running applications on WebSphere Application Server that you want to monitor.
2. On each of these computers, install WASTT.

3. When installation is complete, configure WASTT.
4. Restart each WebSphere Application Server.

Supported products

WASTT supports transaction tracking on WebSphere Application Server.

WASTT also supports tracking to and from WebSphere MQ.

For a list of supported versions:

1. Link to the required ITCAM for Transactions version from ITCAM for Transactions on Documentation Central
2. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**.
3. In the System requirements and prerequisites page, select Transaction Tracking from the Compatible software list.

Supported operating systems

For information about supported operating systems:

1. Link to the required ITCAM for Transactions version from ITCAM for Transactions on Documentation Central
2. In the navigation pane, select **Composite Application Manager for Transactions** > **Prerequisites**.
3. In the System requirements and prerequisites page, select Transaction Tracking from the Supported operating systems list.

Installing WASTT

Install WASTT on all computers running WebSphere Application Server whose transactions you want to track.

Before you begin

WASTT is included in the Transaction Tracking V7.1.0.2 and later packages and is installed using a separate installer. The installer is named `wastt-`*`operating system-bits-vbuildnumber`*.

For information about obtaining the latest version, see the Download information on the ITCAM for Transactions Information Center.

WASTT contains the following components:

- TTARM - Transaction Tracking ARM implementation.
- TTAPI4J -Transaction Tracking API for Java.
- Default XML files for common configuration. The same versions are distributed with the Transaction Collector.
- A WebSphere Application Server `wsadmin` configuration script.

For each platform, there is an installer that includes the 32-bit version native libraries, and an installer that includes 64-bit version native libraries. The configuration script sets the library path of the WebSphere Application Server to the relevant location.

Ensure that you have read and understood the prerequisites information described in "Preparing WebSphere Application Server Transaction Tracking (WASTT)" on page 292.

Procedure

To install WASTT:

1. Extract the WASTT installer, `wastt-`*`operating system-bits`*`-v`*`buildnumber`*, from the Transaction Tracking DVD.
2. Run the installer as an administrator.
3. On the **Welcome** window, click **Next**.
4. Enter the path to the directory where the files will be installed, for example, `C:\IBM\ITM\TMAITM6\ktj\wastt`. This directory is referred to as *wastt.install.dir* in this documentation.

   Note: If you want to use the Application Management Configuration Editor for WASTT, ensure that you install WASTT to the default location.
5. Click **Next** to begin the installation.
6. Check for errors and click **Finish** to complete the installation.

What to do next

Now configure WASTT using the `wsadmin` tool.

Configuring WASTT

Configure WebSphere Application Server properties for WASTT by running the `wsadmin` script, called `configure_wastt.py`, included in the installation. The script configures the default configuration properties and library paths in a WebSphere Application Server profile, and configures the PMI Request Metrics settings to enable ARM.

Before you begin

Ensure that WASTT is installed.

About this task

The `configure_wastt.py` wsadmin script is installed with WASTT. Call the script by running the following configuration script that is installed to *wastt.install.dir*:

- On UNIX systems, `configure.sh`
- On Windows systems, `configure.cmd`

The configuration script sets the logging levels and the log file location for WASTT. WASTT logs `ERROR (COMP:arm ALL)` and log files are stored in the WebSphere Application Server log directory.

Because PMI Request Metrics are configured across profiles, all servers are configured by the configuration script provided with WASTT. The configuration script sets the PMI Request Metrics for WASTT to `HOPS`.

Note: To monitor EJB transactions with WASTT, change the PMI Request Metrics level from `HOPS` to `Performance_debug`.

Procedure

To configure WASTT:

1. If required, set the Transaction Collector address. Open `configure_wastt.py` in a text editor and change the value of the **TRANSACTION_COLLECTOR_ADDRESS** variable. A default value is used if you do not set a Transaction Collector address.
2. On UNIX systems, run the following command from the command-line interface:

   `./configure.sh` *path-to-WAS-profile*
3. On Windows systems, run the following command from the command-line interface:

   `configure.cmd` *path-to-WAS-profile*
4. Repeat for each WebSphere Application Server to which WASTT is installed.
5. To run WebSphere Application Server with WASTT enabled, all required libraries must be in the system library search path:
   a. Run the following commands to add the path to the system library search path:
      - On Linux or Solaris systems, run `LD_LIBRARY_PATH=/opt/IBM/ITM/platform/tj/wastt:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH`. Also run `CANDLE_HOME=/opt/IBM/ITM; export CANDLE_HOME`.
      - On AIX systems, run the above commands for `LIBPATH` instead of `LD_LIBRARY_PATH`.
      - On HP-UX systems, run the above commands for `SHLIB_PATH` instead of `LD_LIBRARY_PATH`.
      - On Windows systems, append `%CANDLE_HOME%/TMAITM6/ktj/wastt` to the `Path` environment variable. `CANDLE_HOME` is set automatically by the IBM Tivoli Monitoring installer.
   b. Restart the WebSphere Application Server.

What to do next

If required, you can modify the address of the Transaction Collector after you have configured WASTT. Using the WebSphere Application Server Admin console, modify the Java system property, **com.ibm.tivoli.tt.collector**. Ensure that you restart WebSphere Application Server for the changes to take effect.

The default profile `All Maximo URLs via ARM` is included in V7.2 and later. This profile extracts Maximo-specific request parameters such as `target id` and uses them to enrich the transaction names. This enables the identification of different transaction types within Maximo®.

Construct additional WASTT transactions that you want to monitor by using the Application Management Configuration Editor. See "Configuring your environment with the Application Management Configuration Editor" in the *Administrator's Guide* for further information.

For WASTT V7.2.0.1 and later, you can enable KBB RAS1 logging in WebSphere Application Server. To enable KBB RAS1 logging when you want to debug a problem:

1. In the WebSphere Administrative Console, select **Application servers** > **server1** > **Process Definition** > **Environment Entries**.

2. Set the property `CYTA_LOGGER_DISABLE=0`
3. For performance reasons, when you have finished debugging, set the property `CYTA_LOGGER_DISABLE=1` to disable KBB RAS1 logging.

Further considerations for WebSphere MQ transaction tracking

Further configuration is required to enable WASTT to track WebSphere MQ XA transactions. No further configuration is required to track non-XA transactions.

Enabling transaction tracking to and from WebSphere MQ for XA transactions

To track WebSphere MQ XA transactions in WebSphere Application Server:

1. Set the **QmgrSyncpointFilter** parameter in the MQ API exits configuration file, *platform*/th/config/ttdcmqexits.cfg, to the name of the queue manager that you want to monitor. For example, `QmgrSyncpointFilter = QMA,QMB`.
2. Restart the queue manager.
3. Restart WebSphere Application Server.

Removing WASTT configuration

Before you uninstall WASTT, you need to first remove its configuration.

Procedure

To remove the configuration for WASTT:

1. On UNIX systems, run the following command from the command-line interface:

   `./unconfigure.sh` *path-to-WAS-profile*
2. On Windows systems, run the following command from the command-line interface:

   `unconfigure.cmd` *path-to-WAS-profile*
3. Restart the WebSphere Application Server.
4. Repeat for each WebSphere Application Server to which WASTT is installed.

Tracking web servers

You can track web servers using one of two methods: Web Response Time or ARM.

Consider the following information when deciding whether to use ARM or Web Response Time to track web servers:

- If you use Web Response Time, continue using the Web Response Time monitoring agent for tracking.
- If your end-to-end composite application supports ARM, use ARM.

  For example, to track from Rational Performance Tester to IBM HTTP Server to WebSphere Application Server, use ARM in all components.
- If your environment contains domains to track that do not support ARM, use Web Response Time.

Enabling ARM transactions

If the application that you want to monitor is not one of the supported domains, but it is an ARM-instrumented application you can still monitor it using ITCAM for Transactions. However, before you can monitor ARM transactions some customization may be necessary after you have installed the Transaction Tracking or Response Time components.

Support for ARM-instrumented applications is available in ITCAM for Transactions V7.1 and later.

The Transaction Collector agent supports monitoring ARM-instrumented applications. Installing Transaction Collector automatically installs support for all ARM-instrumented applications.

The Transaction Collector replaces the V6.2 Client Response Time agent for monitoring ARM applications. If you require Client Response Time workspaces in addition to the Transaction Collector workspaces, you can configure the Client Response Time agent to also monitor ARM.

Table 60 lists the intra-domain and inter-domain tracking supported by Transaction Tracking for the ARM domain.

Table 60. ARM - domain interactions supported

Domain interactions tracked	Notes
ARM to and from ARM	
ARM to WebSphere Application Server	Using ITCAM for Application Diagnostics
ARM to WebSphere Application Server	Using WASTT
ARM to web servers	• IBM HTTP Server • On Demand Routing (ODR) • WebSEAL

Enabling ARM transactions:

To enable ARM transactions:

1. Ensure that you understand the ARM data collection process.
2. Check the general requirements.
3. Configure ARM.
4. Configure the ARM-instrumented application, for example, WebSphere.

ARM data collection process

The ARM data collected is determined by the profiles described in the Application Management Configuration Editor.

These profiles are stored on the computer where the Application Management Configuration is running, and are transferred to the Transaction Collectors periodically. The ARM configuration is transferred to the local disk in the `$CANDLE_HOME/tmaitm6/camconfig/TU/` directory. The ARM library can then read the configuration files and update itself periodically, and turn the ARM data collected into Transaction Tracking API events according to the configuration files. The ARM libraries then send the Transaction Tracking API events to the Transaction Collector defined in the `armconfig.xml` file.

ARM configuration information is logged in the following files:

- *computer name*_tu_*number*.log - standard (RAS1) Transaction Collector log, default location $CANDLE_HOME/logs.

Review the logs at C:\IBM\ITM\logs to determine when a profile was loaded and where it was loaded from.

A component called configdepotmanager handles profile transfer. Use the term *configdepotmanager* in the *computer name*_tu_*number*.log to filter log information so you see only those entries relevant to profile transfer.

Example

Filtering on *configdepotmanager* in the Transaction Collector RAS1 log results in the following types of log messages:

```
1. (4868A78E.0000-788:configdepotmanager.cpp,304,"ConfigDepotManager")
ProductCode=TU
2. (4868A78E.0001-788:configdepotmanager.cpp,291,"init") OriginNode=WINXP-IHS-ITM:
TU ServerRelativeDepot=camconfig/ LocalMasterFile=C:\ibm\ITM/tmaitm6//camconfig/TU/
TU_master.xml ServerMasterFile=camconfig/TU_master.xml
3. (4868A7CA.0000-788:configdepotmanager.cpp,338,"ConfigDepotManager") Found T3
Node:winxp-ihs-itm:T3
4.(4868ADE9.0000-788:configdepotmanager.cpp,403," ConfigDepotManager") Getting new
version of file:MyListener2.xml
5. (4868ADE9.0001-788:configdepotmanager.cpp,403,"ConfigDepotManager") Getting new
version of file:Tracking_Defaults.xml
```

In the previous messages, message:

- *2* shows where the Transaction Collector puts the configuration file for use in the ARM library.
- *3* shows from where the T3 agent has retrieved the configuration files.
- *4* and *5* show when a new set of configuration files are retrieved by the Transaction Collector.

Note: The above sample shows the raw log. When viewed using the RAS1 log viewer, the number before the first colon (:) is a timestamp.

General requirements

Before configuring WebSphere and DB2 to issue ARM calls, ensure that you have checked here first.

Required software

ARM-supported applications:

- DB2
- IBM HTTP Server
- On Demand Routing (ODR)
- WebSEAL
- WebSphere Application Server
- Robotic Response Time (T6)

See *Compatible Software* prerequisite information in the required version of ITCAM for Transactions Information Center on Documentation Central for further information.

Native ARM libraries

All native libraries required by ARM can be found in the `tusupport` directory, which is created when the Transaction Collector agent is installed. The subdirectories `32` and `64` contain the 32-bit and 64-bit libraries respectively.

- For Windows systems, `%CANDLE_HOME%\tmaitm6\tusupport\32|64`
  For example, `c:\ibm\ITM\TMAITM6\tusupport\32`.
- For UNIX systems, `$CANDLE_HOME/platform/tu/tusupport/32|64`

Note: When installing the Transaction Collector on a 64-bit platform, the ARM libraries that are put in the $CANDLE_HOME/platform/tu/tusupport folder are 64-bit. When using these libraries in an application, you need the 64-bit TTAPI library. This is available in the *platform*/tu/tusupport/sdk/*platform*-64.tar.gz package. Alternatively, configure the ARM-instrumented application to use the subdirectory `64`, which contains both the 64-bit ARM and TTAPI libraries.

Table 61 lists the libraries that must be in the library path of the ARM-instrumented application.

Table 61. Native libraries required by ARM

Platform	Library files
Windows 32-bit platforms	• `libarm32.dll` - ARM 2 only • `libarm4.dll` • `ttapi.dll` • `pthread.dll` • `kbb.dll` (optional, used for RAS1 logging)
Windows 64-bit platforms	• `libarm_64.dll` - ARM 2 only • `libarm4_64.dll` • `ttapi.dll` • `pthread.dll` • `kbb.dll` (optional, used for RAS1 logging)
AIX 32-bit and 64-bit platforms	• `libarm.a` - ARM 2 only • `libarm4.a` (archive containing both 32-bit and 64-bit shared objects) • `libttapi.a` • `libkbb.a` (optional, used for RAS1 logging)
All other UNIX and Linux 32-bit platforms	• `libarm.so` - ARM 2 only • `libarm4.so` • `libttapi.so` • `libkbb.so` (optional, used for RAS1 logging) **Note:** substitute `.sl` for `.so` on HP-UX.
All Other UNIX and Linux 64-bit platforms	• `libarm_64.so` - ARM 2 only • `libarm4_64.so` • `libttapi.so` • `libkbb.so` (optional, used for RAS1 logging) **Note:** substitute `.sl` for `.so` on HP-UX

For Java applications, the following jar files must be included in the application's classpath:

- armjni.jar - ARM 2 only
- armjni4.jar

Configuration files

Both ARM and filter configuration files must be installed on the computer whose web server you are monitoring. If the Transaction Collector is installed remotely, you may need to add these files manually.

armconfig.xml

If your installation does not include an ARM configuration file you can create an XML file named armconfig.xml and add it to $CANDLE_HOME/tmaitm6/arm/.

For versions of ITCAM for Transactions up to and including V7.1.0.2 add the following lines to armconfig.xml:

```
<configuration>
 <performancelogging>-1</performancelogging>
   <resetperfonlog>false</resetperfonlog>
   <traceenabled>true</traceenabled>
   <filebuffersize>32767</filebuffersize>
   <queuesize>10</queuesize>
   <perfdetaillevel>0</perfdetaillevel>
   <ttconnectionstring>tcp:TU_host:TU_port</ttconnectionstring>
</configuration>
```

Note: When traceenabled is set to true, logging is enabled and messages are saved to /opt/IBM/ITM/tmaitm6/logs/trace-armdebug.log.

For newer versions of ITCAM for Transactions from V7.2 onwards, add the following lines to armconfig.xml:

```
<configuration>
 <performancelogging>-1</performancelogging>
 <resetperfonlog>false</resetperfonlog>
 <filebuffersize>32767</filebuffersize>
 <queuesize>10</queuesize>
 <perfdetaillevel>0</perfdetaillevel>
 <ttconnectionstring>tcp:TU_host:TU_port</ttconnectionstring>
</configuration>
```

Ensure you add the location of the remote Transaction Collector to ttconnectionstring. For IPv6, use the format tcp:[*TU_host*]:*TU_port*, for example, tcp:[::1]:5455.

Note: In newer versions of ITCAM for Transactions, traceenabled is replaced by the KBB_RAS1_LOG and KBB_RAS1 environment variables. These environment variables enable KBB logging. If a process such as a web server creates a child process, the environment variables may not be inherited. To enable logging in this situation, create a file named KBBENV in the current working directory of the process and add the following lines:

```
KBB_RAS1_LOG=path to log files
KBB_RAS1=ERROR (COMP:ARM ALL)
```

Tracking_defaults.xml

Filter configuration files are created by the Application Management Configuration Editor (T3 agent) and transmitted to the Transaction Collector via the Tivoli Enterprise Monitoring Server. The Transaction Collector saves the files locally to the computer on which it is installed. The ARM DLL loads the filter configuration files when there are changes.

If the Transaction Collector and web server are installed on different computers, the ARM DLL is therefore also remote from the Transaction Collector and cannot access the filter configuration files. You must either install the Transaction Collector on the computer with the web server, or copy the `*_Tracking_Defaults.xml` files to that computer manually.

The location of the filter configuration files read by the ARM library is determined by the value of the `CANDLE_HOME` environment variable, and whether the ARM library is loaded by the Robotic Response Time (T6) agent:

- If the ARM library is loaded by the Robotic Response Time agent, the ARM library reads the XML files in `$CANDLE_HOME/tmaitm6/camconfig/T6`
- Otherwise the ARM library reads the XML files in both:
  - `$CANDLE_HOME/tmaitm6/camconfig/T4`
  - `$CANDLE_HOME/tmaitm6/camconfig/TU`

Enabling ARM

The steps required to enable ARM depend on whether the server on which ARM is installed also has aTransaction Collector installed.

Enabling ARM on a server with the Transaction Collector installed

The Transaction Collector installs the files required for enabling ARM.

These files include the libraries and the ARM configuration file (armconfig.xml). Filter files created by the Application Management Configuration Editor when you apply filters to a profile are also automatically loaded by the ARM DLL.

For most systems, you should now only need to configure the ARM-instrumented application, such as WebSphere and associated web servers.

Enabling ARM on WebSphere

You can configure WebSphere to use the ARM library and send events to Transaction Tracking without having to instrument Transaction Tracking for your web application.

Tip: You can use WASTT to monitor WebSphere Application Server instead of configuring WebSphere separately to use ARM. If you use WASTT, the required variables and directories in WebSphere Application Server are configured automatically.

Configuring WebSphere

You must set properties and Request Metrics before ARM is enabled on WebSphere. To set the properties and metrics:

1. Using the administrative console on WebSphere, set the Request Metrics. Select **Monitoring and Tuning** > **Request Metrics**. Set the following in each section:
   a. Select **Prepare Servers for Request metrics collection**.

b. **Components to be instrumented**, select **All**.
c. **Trace level**. Select one of the following:
   - **Hops** - entry and exit only. Gives overall transaction performance. Hops is the preferred option.
   - **Performance_Debug** - Greater detail including session beans.
   - **Debug** - Most detail including entity bean get and set timings.
d. **Request Metrics Destination** to the ARM agent:
   1) Select **Application Response Measurement (ARM) agent**.
   2) Set **Agent type** to `ARM40`.
   3) Set **ARM transaction factory implementation class name** to `com.ibm.tivoli.tt.ArmTransactionFactory`.

2. Using the administrative console, set the custom properties:
   a. Select **Application servers** > **server1** > **Java and Process Management** > **Process Definition** > **Java Virtual Machine** > **Custom Properties**.
   b. Set the `ws.ext.dirs` property to the following:
      - For UNIX systems, `/opt/IBM/ITM/`*`platform`*`/tu/tusupport`
      - For Windows systems, `C:\IBM\ITM\TMAITM6\tusupport`
   c. Set `java.library.path`:
      - For UNIX systems, depending on your operating system, either `/opt/IBM/ITM/tmaitm6/`*`platform`*`/tu/tusupport/32` or `/opt/IBM/ITM/tmaitm6/`*`platform`*`/tu/tusupport/64`
      - For Windows systems, depending on your operating system, either `C:\IBM\ITM\TMAITM6\tusupport` or `C:\IBM\ITM\TMAITM6\tusupport\64`
3. To see the ARM logs when the ARM library is used by WebSphere Application Server, use the administrative console and select **Application servers** > **server1** > **Process Definition** > **Environment Entries**, and set the properties:

   ```
   KBB_RAS1=ALL
   KBB_RAS1_LOG=log file
   CYTA_LOGGER=ttapi
   ```

4. If you are using DB2 and transactions are received from the WebSphere Application Server, set the following custom property to true: **com.ibm.websphere.pmi.reqmetrics.PassCorrelatorToDB**. If the custom property is not in the list, create a new property: enter the property name in the **Name** field, and a valid value in the **Value** field.
5. Restart WebSphere.

   WebSphere now issues ARM calls and passes correlators to DB2.

Enabling ARM on web servers

After you have configured ARM on WebSphere, deploy the ARM configuration to the web server so that it can issue ARM calls.

Note: Before starting your web server, ensure that Transaction Collector is installed, either locally or remotely (see Chapter 8, “Installing Transaction Tracking,” on page 187) and the ARM environment is distributed (see “ARM data collection process” on page 297).

Enabling ARM on a server separate to the Transaction Collector

If the server on which you want to enable ARM is separate to the Transaction Collector, additional steps are required to enable ARM.

To enable ARM data to be sent to a remote Transaction Collector, complete the following steps:

1. Copy the native libraries from the computer on which the Transaction Collector is installed to that on which the ARM-instrumented application is installed. Append the Transaction Tracking library path to the existing `LD_LIBRARY_PATH` ensuring that you do not overwrite any other application paths.

   For versions of Transaction Tracking earlier than V7.2.0.1, do one of the following:

   - Uncompress the TTAPI SDK from `/tusupport/ttapi/sdk/` to `tusupport/32|64`
   - Point `LD_LIBRARY_PATH` to `tusupport/32|64`

   See "Native ARM libraries" on page 299 for further information.
2. Copy `armconfig.xml` from the computer on which the Transaction Collector is installed to the computer on which the ARM-instrumented application is installed and add the location of the remote Transaction Collector to the file. See "Configuration files" on page 300 for further information.
3. Restrict resource usage if required. For example, on Linux systems, set the following limits:
   - ulimit -s 128
   - ulimit -n 10240
4. Synchronize the filter configuration files from the Application Management Configuration Editor as required. See "Configuration files" on page 300 for further information.

Enabling ARM on WebSEAL

To enable ARM on WebSEAL follow the procedure described here.

To enable ARM on WebSEAL:

1. Ensure that you have the following libraries:

   `libarm4.so`

   `libttapi.a`

   `libkbb.a` for RAS1 logging
2. Set the following values in the WebSEAL configuration file, *WEBSEAL_HOME*`/etc/webseald-`*instance*`.conf`:

   ```
   library = /opt/IBM/ITM/aix533/tu/tusupport/libarm4.so
   ```

   If you are using RPT with WebSEAL, `accept-correlators = yes`

   ```
   enable-arm = yes
    insert-client-real-ip-for-option-r = yes
   report-transactions = yes
   app-group = webseal
   app-instance = WebSEAL_iago
   ```
3. Set the library paths in the `pdweb_start` file, `/opt/pdweb/bin/pdweb_start`. Add the following environment variables to the beginning of the `pdweb_start` file:

   ```
   CANDLE_HOME=/opt/IBM/ITM
   export CANDLE_HOME
   ```

```
LIBPATH=$CANDLE_HOME/aix533/tu/tusupport/32:$LIBPATH
LIBPATH=$CANDLE_HOME/tmaitm6/aix523/lib:$LIBPATH
export LIBPATH

KBB_RAS1=ALL
KBB_RAS1_LOG=/tmp/pdweb.ras1
export KBB_RAS1
export KBB_RAS1_LOG
```

Note: On AIX systems, the `webseald` program may have the `setuid` security feature which stops `LIBPATH` being propagated. Run `pdweb_start` as `root` to circumvent this problem.

4. Restart WebSEAL. The RAS1 log file, `/tmp/pdweb.ras1` defined in the previous step is generated immediately if ARM has been successfully enabled.
5. If required, modify the location of the remote Transaction Collector in the `armconfig.xml` file to which the ARM data will be sent.
6. If required, customize filters for WebSEAL data in the Application Management Configuration Editor.

System-specific configuration

Different systems may require different configurations to enable tracking for ARM-instrumented applications. Some of these configurations are discussed in this section.

Enabling ARM on IBM HTTP Server when using ITCAM for Application Diagnostics on WebSphere Application Server

To use ITCAM for Application Diagnostics as the data collector on WebSphere Application Server and ARM as the data collector on IBM HTTP Server, do not enable ARM on WebSphere Application Server, instead enable ARM on IBM HTTP Server.

Procedure

To enable ARM on IBM HTTP Server:

1. Export *CANDLE_HOME*. Run *CANDLE_HOME*=/opt/IBM/ITM; export *CANDLE_HOME*. The location of *CANDLE_HOME* is determined by where you installed the Transaction Collector. On Windows systems, *CANDLE_HOME* is set automatically by the IBM Tivoli Monitoring installer. See "General requirements" on page 298 for location information.
2. Set the environment variables:
   - On Linux or Solaris systems, run `LD_LIBRARY_PATH= /opt/IBM/ITM/`*platform*`/tu/tusupport:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH`.
   - On AIX systems, run the above commands for `LIBPATH` instead of `LD_LIBRARY_PATH`.
   - On Windows, append *%CANDLE_HOME%*`/TMAITM6/tusupport` to the Path environment variable. *CANDLE_HOME* is set automatically by the IBM Tivoli Monitoring installer.
3. If you are using a remote Transaction Collector, modify `armconfig.xml` to point to the remote Transaction Collector. See "Updating the location of a remote Transaction Collector" on page 307 for further information.
4. Modify *IHS_DIR*`/Plugins/config/webservername/plugin-cfg.xml` and set the parameter **`armenabled=true`**.
5. Restart IBM HTTP Server.

What to do next

Tip: If required later, you can disable ARM. Set **armenabled=false** and restart IBM HTTP Server.

Running ARM-instrumented applications on UNIX systems

Transaction Tracking provides a shell script that allows you to modify the environment on UNIX operating systems so that applications that are ARM-instrumented can make ARM calls and find the appropriate runtime libraries. The shell script is named `setup4ARM.sh`; it is placed in the `$CANDLE_HOME/arch/tu/tusupport` directory during installation. To use this script, you must provide a single parameter to the script that is in the path to the script. For example, `../setup4ARM.sh ./setup4ARM.sh`.

Enabling ARM on HP-UX systems

The ARM support files for HP-UX systems are installed with MQ Tracking. To enable ARM on HP-UX systems, first install MQ Tracking and then configure the support files.

Before you begin

Install MQ Tracking, see "Installing MQ Tracking on Windows, Linux, and UNIX systems" on page 224 for further information.

Procedure

To configure ARM on HP-UX systems, complete the following steps after you have installed MQ Tracking:

1. Copy the ARM configuration files from the sample: `cp -r /opt/IBM/ITM/hp11/th/arm/example_camconfig/TU /opt/IBM/ITM/tmaitm6/camconfig/TU`
2. Create a new directory for the ARM files: `mkdir /opt/IBM/ITM/tmaitm6/arm`
3. Add the following lines to `armconfig.xml`:

```
<configuration>
 <performancelogging>-1</performancelogging>
 <resetperfonlog>false</resetperfonlog>
 <filebuffersize>32767</filebuffersize>
 <queuesize>10</queuesize>
 <perfdetaillevel>0</perfdetaillevel>
 <ttconnectionstring>tcp:TU_host:TU_port</ttconnectionstring>
</configuration>
```

4. In `armconfig.xml`, add the following value for `ttconnectionstring`:

   For IPv4, `tcp:TU_host:TU_port`

   For IPv6, `tcp:[TU_host]:TU_port`

   where *TU_host* is the host name of the Transaction Collector, and *TU_port* is the port on which the Transaction Collector is listening for data. For example:

   For IPv4, `<ttconnectionstring>tcp:127.0.0.1:5455</ttconnectionstring>`

   For IPv6, `<ttconnectionstring>tcp:[::1]:5455</ttconnectionstring>`

Tracking DB2 systems

For best results, use the Transaction Tracking JDBC feature from ITCAM for Application Diagnostics (was ITCAM for WebSphere) rather than ARM to track DB2.

By using ITCAM for Application Diagnostics (was ITCAM for WebSphere) V6.1 FP4 or higher you can track indicative response times for DB2.

Only registered JDBC data source calls are supported for WebSphere to DB2 transactions. Direct DB2 database access, such as Java code directly communicating with DB2 is not supported.

DB2 only issues ARM calls for sub-transactions when they originate from WebSphere. Therefore you can only get ARM data from DB2 for WebSphere transactions, and then only if WebSphere is configured to send the ARM correlator to DB2. See "Enabling ARM on WebSphere" on page 301 for further information about configuring WebSphere to issue ARM calls and pass its correlator to DB2.

DB2 registers with ARM if it finds `libarm4.dll` in its library path. In order to issue start or stop calls for transactions, DB2 requires a correlator to be passed in with the JDBC connection (configured in WebSphere). To configure the DB2 library path, run the following command from within the DB2 environment:

```
db2set DB2LIBPATH=libarm4.dll directory location
```

where *libarm4.dll directory location* is:

- AIX - `$CANDLE_HOME/tmaitm6/aix533/lib/`
- HPUX - `$CANDLE_HOME/hp11/tu/tusupport/:$CANDLE_HOME/hp11/tu/lib/ :$CANDLE_HOME/tmaitm6/hp11/lib/`
- Linux - `$CANDLE_HOME/li6263/tu/tusupport:$CANDLE_HOME/li6263/tu/lib/ :$CANDLE_HOME/tmaitm6/li6263/lib`
- Solaris - `$CANDLE_HOME/sol293/tu/tusupport/:$CANDLE_HOME/sol293/tu/lib/ :$CANDLE_HOME/tmaitm6/sol283/lib/`

Note: On Windows 64-bit systems also add the value of `DB2LIBPATH` to the system `Path` environment variable.

Ensure that you restart the computer.

`$CANDLE_HOME` defaults to `c:\ibm\itm` for Windows systems and `/opt/IBM/ITM/` for UNIX systems.

Note: On Windows systems, IBM Tivoli Monitoring automatically includes `c:\ibm\itm\tmaitm6\` in the system path.

Note: If you use 64-bit ARM with 64-bit DB2, you must manually rename the 64-bit library `libarm4_64.a` to `libarm4.a` to ensure `libarm4.a` becomes a 64-bit library.

Tracking ARM data for ITCAM for Application Diagnostics

If you enable both ITCAM for Application Diagnostics and ITCAM for Transactions to monitor ITCAM for Application Diagnostics and report ARM data, you may get duplicate data.

Duplicate data may appear in the Tivoli Enterprise Portal if you set both of the following options using the Data Collector Configuration Tool:

- Data collection for ITCAM for WebSphere's Tivoli Enterprise Portal interface
- Data collection for ITCAM for Transactions' Tivoli Enterprise Portal interface

To resolve this problem, reconfigure the data collectors to report only to the ITCAM for Application Diagnostics agent instead of to both ITCAM for Application Diagnostics and ITCAM for Transactions. Ensure that you restart the WebSphere Application Servers.

Updating the location of a remote Transaction Collector

You may want to change the location of the Transaction Collector to which ARM data is sent. Use the ARM configuration file, `armconfig.xml`, to modify the location of a remote Transaction Collector.

From Transaction Tracking V7.1.0.1 onwards, `armconfig.xml` is included in the Transaction Tracking installation at `$CANDLE_HOME/tmaitm6/arm/`.

To update the location of a remote Transaction Collector:

1. Open `armconfig.xml` in a text editor.
2. Update **`ttconnectionstring`** with the location of the remote Transaction Collector using the format:

   For IPv4, `tcp:`*`TU_host`*`:`*`TU_port`*

   For IPv6, `tcp:[`*`TU_host`*`]:`*`TU_port`*

   where *TU_host* is the host name of the Transaction Collector, and *TU_port* is the port on which the Transaction Collector is listening for data. For example:

   For IPv4, `<ttconnectionstring>tcp:127.0.0.1:5455</ttconnectionstring>`

   For IPv6, `<ttconnectionstring>tcp:[::1]:5455</ttconnectionstring>`
3. Restart the application that loads the ARM library.

Chapter 11. Working remotely

IBM Tivoli Monitoring provides you with the ability to deploy resource monitoring across your environment from a central location, the monitoring server. You can use the remote deployment feature to deploy and configure monitoring agents, to deploy maintenance and upgrades to agents, and to start and stop agents.

Note: For complete information about remote deployment, see chapter *Deploying monitoring agents across your environment* in the *IBM Tivoli Monitoring Installation and Setup Guide*.

Table 62 describes the steps required to set up and manage remote agent deployment:

Table 62. Remote agent deployment tasks

Goal	Where to find information
Create and populate the agent deploy depot with installable agent images	"Populating the agent depot" on page 310
Manage the agent depot	"Managing your agent depot" on page 314
Use one agent depot for all the monitoring servers in your monitoring environment.	"Sharing an agent depot across your environment" on page 314
Deploy OS agents	"Deploying OS agents" on page 315
Deploy non-OS agents	"Deploying non-OS agents" on page 317
Upgrade non-OS agents remotely	"Upgrading non-OS agents remotely" on page 319
Remove non-OS agents remotely	"Removing non-OS agents remotely" on page 319

Note: After you have deployed an agent, you can reconfigure it. Right-click on the agent in the Tivoli Enterprise Portal and select **Configure**. For more information about post-configuration tasks, see the section *Post-installation configuration and customization* in the *IBM Tivoli Monitoring Installation and Setup Guide*.

You can also use the remote agent deployment function to configure deployed agents and install maintenance on your agents. For information, see the *IBM Tivoli Monitoring Administrator's Guide*. See the *IBM Tivoli Monitoring Command Reference* for commands that you can use to perform these tasks.

Important:

> Run the **tacmd login** command before executing commands from the tacmd library. This requirement does not apply to the addBundles command. Run the **tacmd logoff** command after you finish using the tacmd command library.

Note: If you install an agent manually and then perform a remote uninstall, the remote uninstall does not remove the entry from Windows Add/Remove program. You must remove the entry manually.

Populating the agent depot

The *agent depot* is an installation directory on the monitoring server from which you deploy agents and maintenance packages across your environment. Before you can deploy any agents from a monitoring server, you must first populate the agent depot with bundles. A bundle is the agent installation image and any prerequisites.

When you add a bundle to the agent depot, you need to add the bundle that supports the operating system to which you want to deploy the bundle. For example, if you have a Windows Tivoli Enterprise Monitoring Server and Web Response Time agent on Linux and you want to remote configure the Linux agent, the Tivoli Enterprise Monitoring Server must have the Web Response Time Linux package in the depot. (If you are installing from different media for each platform type, for example, Windows, AIX and Solaris, HP-UX, Linux, you need to add the bundle from the specific platform media for the component.)

You can have an agent depot on each monitoring server in your environment or share an agent depot, as described in "Sharing an agent depot across your environment" on page 314. If you choose to have an agent depot for each monitoring server, you can customize the agent depot based on the types of bundles that you want to deploy and manage from that monitoring server. For example, if you have a monitoring server dedicated to monitoring with Web Response Time agents, populate the depot with Web Response Time-related agent bundles. If you deploy an agent from a remote monitoring server, you must have a agent bundle in the depot available to the monitoring server.

Note: Agent depots cannot be located on a z/OS monitoring server.

There are two methods to populate an agent depot:

- Populating the agent depot from the installation image
  - "Populating the agent depot during installation: Windows" on page 311
  - "Populating the agent depot during installation: Linux and UNIX" on page 312
- "Populating an agent depot with the tacmd addBundles command" on page 313

You can use the installation image to populate the agent depot only when you are populating the depot with bundles for the same operating system as your monitoring server. For example, you can use the installation image to add a bundle for a Windows agent to a Windows monitoring server, but you cannot use the Linux installation image to add a Linux bundle to a Windows monitoring server. If you need to add bundles for operating systems other than that used by your monitoring server, use the `tacmd addBundles` command, as described in "Populating an agent depot with the tacmd addBundles command" on page 313.

Note:

Only Tivoli-provided product agent bundles should be loaded into the IBM Tivoli Monitoring deployment depot. User-provided or customized bundles are not supported. Use only Tivoli provided tacmd commands to process bundles and to execute agent deployments. Manual manipulation of the depot directory structure or the bundles and files within it is not supported and may void your warranty.

Populating the agent depot during installation: Windows

The procedure to populate the agent depot from the Windows installation image differs based on the installation image (base IBM Tivoli Monitoring or application agent) that you are using. Use the procedure in this section that applies to the image you are using:

- Base IBM Tivoli Monitoring installation image
- Application agent installation image

Base IBM Tivoli Monitoring installation image:

Use the following steps to populate the agent depot from the IBM Tivoli Monitoring installation image:

1. Launch the installation wizard by double-clicking the `setup.exe` file in the `\Windows` subdirectory of the installation image.
2. Click **Modify** on the Welcome window and select **Next**.
3. Click **OK** the warning message regarding existing components on this computer.
4. Click **Next** on the Add or Remove Features window without making any changes. (Do not clear any selected items because this removes them from the computer.)
5. On the Agent Deployment window, select the agents that you want to add to the depot and click **Next**.
6. Review the installation summary and click Next to begin the installation. After the agents are added to the agent depot, a configuration window (called the Setup Type window) is displayed.
7. Clear all selected components. You have already configured all components on this computer and do not need to reconfigure any now. Click **Next.**
8. Click **Finish** to complete the installation.
9. Click **Finish** on the Maintenance Complete window.

Application agent installation image:

Use the following steps to populate the agent depot from an application agent installation image:

1. Launch the installation wizard by double-clicking the `setup.exe` file in the `\Windows` subdirectory of the installation image.
2. Click **Next** on the Welcome window.
3. Click **Next** on the Select Features window without making any changes.
4. On the Agent Deployment window, select the agents that you want to add to the depot and click **Next**.
5. Review the installation summary and click **Next** to begin the installation. After the agents are added to the agent depot, a configuration window (called the Setup Type window) is displayed.
6. Clear all selected components. You have already configured all components on this computer and do not need to reconfigure any now. Click **Next.**
7. Click **Finish** to complete the installation.
8. Click **Finish** on the Maintenance Complete window.

Populating the agent depot during installation: Linux and UNIX

Populating the agent depot during a Linux and UNIX installation.

Use the following steps to populate the agent depot from the Linux or UNIX installation image:

1. In the directory where you extracted the installation files, run the following command:`./install.sh`.
2. When prompted for the IBM Tivoli Monitoring home directory, press Enter to accept the default (`/opt/IBM/ITM`). If you want to use a different installation directory, type the full path to that directory and press Enter.
3. If the directory you specified does not exist, you are asked whether to create it. Type `y` to create this directory.
4. The following prompt is displayed:

   ```
   Select one of the following:
   1) Install products to the local host.
   2) Install products to depot for remote deployment (requires TEMS).
   3) Install TEMS support for remote seeding 4) Exit install.
      Please enter a valid number:

   Type 2 to start the installation and press Enter.
   ```

   The end user license agreement is displayed. Press Enter to read through the agreement.
5. Type 1 to accept the agreement and press Enter.
6. Type the number that corresponds to the agent or agents that you want to add to the agent depot and press Enter. If you are going to add more than one agent, use a comma (,) to separate the numbers. To select all available agents, type `all`. You can select multiple agents with consecutive corresponding numbers by typing the first and last numbers for the agents, separated by a hyphen (-). For example, to add all of the agents between 8 and 12, type 8-12. To clear an agent that you previously selected, type the number for the agent again.

   Note: Use the following keys to navigate the list of agents:

   U Moves up a line in the list.

   D Moves down a line in the list.

   F Moves forward one page in the list.

   B Moves back one page in the list.
7. When you have specified all the agents that you want to add to the agent depot, type E and press Enter to exit.

Populating an agent depot with the tacmd addBundles command

Populating an agent depot with the **tacmd addBundles** command.

To add bundles for operating systems other than that used by your monitoring server, use the `tacmd addBundles` command.

Restriction: Only Tivoli-provided product agent bundles should be loaded into the IBM Tivoli Monitoring deployment depot. User-provided or customized bundles are not supported. Use only Tivoli provided `tacmd` commands to process bundles and to execute agent deployments. Manual manipulation of the depot directory structure or the bundles and files in it is not supported and may void your warranty.

Important: Run the `tacmd login` command before executing commands from the `tacmd` library. Run the `tacmd logoff` command after you finish using the `tacmd` command library.

To populate the agent depot using the `tacmd addBundles` command, run the following command:

```
tacmd addBundles
  [-i IMAGE_PATH]
  [-t PRODUCT_CODE]
  [-p OPERATING_SYSTEM]
  [-v VERSION]
  [-n]
  [-f]
```

For the full syntax, including parameter descriptions, see *IBM Tivoli Monitoring Command Reference*.

Examples:

- The following example copies every agent bundle, including its prerequisites into the agent depot on a UNIX from the installation media (cd image) located at `/mnt/cdrom/`:

  ```
  tacmd addBundles -i /mnt/cdrom/unix
  ```

- The following example copies all agent bundles for the Oracle agent into the agent depot on a UNIX computer from the installation media (cd image) located at `/mnt/cdrom/`:

  ```
  tacmd addBundles -i /mnt/cdrom/unix -t or
  ```

- The following example copies all agent bundles for the Oracle agent into the agent depot on a Windows computer from the installation media (cd image) located at `D:\WINDOWS\Deploy`:

  ```
  tacmd addBundles -i D:\WINDOWS\Deploy -t or
  ```

- The following example copies the agent bundle for the Oracle agent that runs on the AIX version 5.1.3 operating system into the agent depot on a UNIX computer from the installation media (cd image) located at`/mnt/cdrom/`:

  ```
  tacmd addbundles -i /mnt/cdrom/unix -t or -p aix513
  ```

By default, the agent depot is located in the *itm_installdir*`\CMS\depot directory` on Windows and *itm_installdir*`/tables/`*tems_name*`/depot directory` on UNIX. The **tacmd addBundles** command puts the agent bundle in that location unless another location is defined in the monitoring server configuration file for `DEPOTHOME`.

If you want to change this location, do the following before you run the tacmd addBundles command:

1. Open the KBBENV monitoring server configuration file located in the *itm_installdir*/CMS/ directory on Windows and the *itm_installdir*/tables/*tems_name* directory on Linux and UNIX.
2. Locate the DEPOTHOME variable. If it does not exist, add it to the file.
3. Type the path to the directory that you want to use for the agent depot.
4. Save and close the file.
5. On UNIX or Linux only, add the same variable and location to the kbbenv.ini file located in *itm_installdir*config/kbbenv.ini.

   Note: If you do not add the variable to the kbbenv.ini file, it is deleted from the KBBENV file the next time the monitoring server is configured.

Managing your agent depot

Use the following commands to manage your agent depot:

Table 63. Agent depot management commands

Command	Description
tacmd listbundles	Lists the details for one or more bundles available to be added to the local agent depot.
tacmd removebundles	Deletes one or more bundles from the local agent depot.
tacmd viewdepot	Lists the types of bundles available in either the local or remote agent depot.

See the IBM Tivoli Monitoring Command Reference for the full syntax of these commands.

Important: Only Tivoli-provided product agent bundles should be loaded into the IBM Tivoli Monitoring deployment depot. User-provided or customized bundles are not supported. Use only Tivoli provided tacmd commands to process bundles and to execute agent deployments. Manual manipulation of the depot directory structure or the bundles and files within it is not supported and may void your warranty.

Sharing an agent depot across your environment

If your monitoring environment includes multiple monitoring servers (a hub monitoring server and remote monitoring servers), you can put your agent depot in a central location, such as a shared file system, and access the depot from all of the monitoring servers.

After populating your agent depot with either of the methods described in "Populating the agent depot" on page 310, use the following steps to share the agent depot:

1. Open the KBBENV monitoring server configuration file located in the <*itm_installdir*>\CMS directory on Windows and the <*itm_installdir*>/tables/<*tems_name*> directory on Linux and UNIX.
2. Locate the DEPOTHOME variable. By default, the agent depot is located in the <*itm_installdir*>\CMS depot directory on Windows and the <*itm_installdir*>/tables/<*tems_name*> depot directory on UNIX or Linux.

3. Type the path to the shared agent depot for the DEPOTHOME variable.
4. Save and close the file.
5. On UNIX or Linux only, add the same variable and location to the kbbenv.ini file located in `<itm_installdir>/config/kbbenv.ini`.

   Note: If you do not add the variable to the kbbenv.ini file, it will be deleted from the KBBENV file the next time the monitoring server is reconfigured.

If you are using a Windows monitoring server connecting to a depot on another Windows computer, you must set the service ID for the Windows monitoring server to "Administrator." Also, instead of specifying a mapped drive letter for the path to the depot directory, use the UNC path (such as `\\server\share`).

Use the following steps to change the service ID:

1. From the Control Panel, double-click **Administrative Tools**.
2. Double-click **Services** .
3. Right-click **Tivoli Enterprise Monitoring Svcs** and click **Properties**.
4. On the **Log On** tab, select **This Account**.
5. Type `Administrator` in the **This Account** field.
6. Type the password for the administrator in the **Password** field. Confirm the password by typing it again in the **Confirm password** field.
7. Click **Enable**.

   If the Administrator user does not have Logon as a service right, you are prompted to add it.

Deploying OS agents

Before you can deploy any non-OS agent, you must first install an OS agent on the computer where you want the non-OS agent to be deployed. In addition to monitoring base OS performance, the OS agent also installs the required infrastructure for remote deployment and maintenance.

Note: Ensure that you have populated your agent depot, as described in “Populating the agent depot” on page 310, before attempting to deploy any agents.

You can install the OS agent locally, as described in “Installing Response Time monitoring agents and related software” on page 101 or remotely using the tacmd createNode command.

The tacmd createNode command creates a directory on the target computer called the node. This is the directory into which not only the OS agent is installed, but where any non-OS agents are deployed.

The tacmd createNode command uses one of the following protocols to connect to the computers on which you want to install the OS agent:

- Server Message Block (SMB), used primarily for Windows servers.
- Secure Shell (SSH), used primarily by UNIX servers, but also available on Windows.

  Note: Only SSH version 2 is supported.
- Remote Execution (REXEC), used primarily by UNIX servers, but not very secure.

- Remote Shell (RSH), used primarily by UNIX servers, but not very secure.

Requirements for the tacmd createNode command

Before you can use the tacmd createNode command to deploy OS agents, ensure the following:

- On Windows, the user ID that you specify using the -u parameter must have administrator privileges on the target computer. On UNIX and Linux, you must specify the "root" user ID using the -u parameter and the root password using the `-p` parameter for the **tacmd createNode** command to execute correctly. No other user ID may be specified.
- Any computer to which you want to deploy the OS agent must have a supported protocol installed.
- Security in your environment must be configured to permit createNode to pass through the firewall, using the protocol that you specify in the command parameters.
- On Windows computers:
  - SMB requires that the default, hidden, and administrative share are available on the drive being accessed and on the drive that hosts the System temporary directory.
  - SMB signing is not supported when connecting using SMB. The computer to which you are deploying an OS agent cannot require SMB signing.
  - For Windows XP, disable Simple File Sharing. Simple File Sharing requires that all users authenticate with guest privileges. This is not supported for createNode. To disable Simple File Sharing, do the following:
    1. Open the Windows Explorer
    2. Click Tools → **Folder Options**
    3. Click the **View** tab
    4. Scroll through the list of settings to **Use Simple File Sharing**
    5. Clear the check box next to **Use Simple File Sharing** and click **OK**
  - For Windows XP computers with Service Pack 2, disable the Internet Connection Firewall.
  - For Windows XP computers, set Network Access Sharing and Security to "Classic - local users authenticate as themselves." Use the following steps:
    1. From the Control Panel, double-click Administrative Tools.
    2. Double-click **Local Security Policy**.
    3. Expand **Local Policies** and click **Security Options**.
    4. Right-click **Network access**: **Sharing and security for local accounts** and click **Properties**.
    5. Select **Classic - local users authenticate as themselves** from the list and click **OK**.
  - For all Windows computers, enable remote registry administration. (This is enabled by default.)
- On UNIX systems, if you are using the RSH protocol, run the tacmd createNode command as root on the monitoring server.
- If you are deploying the OS agent to a UNIX or Linux computer, that computer must have either the ksh shell. Only the Korn shell is supported for the execution of the installation and runtime scripts.

- If you are using SSH V2 (for either Windows or UNIX), configure SSH on the target computers to permit the use of password authentication. To permit this, do the following:
  1. Edit the `/etc/ss/sshd_config` file on the target computer.
  2. Locate the following line: `PasswordAuthentication .`
  3. Change the `no` to `yes` and save the file.
  4. Restart the daemon

  Note: If you are using private key authentication in your environment, you do not need to set SSH to permit password authentication.

Using the tacmd createNode command

To deploy an OS agent from the command line interface, use **tacmd createNode** command. For the full syntax, including parameter descriptions, see *IBM Tivoli Monitoring Command Reference.*

For example, the following command deploys the UNIX OS monitoring agent on the server1.ibm.com computer in the /opt/IBM/ITM directory. The installation is done as the root user.

```
tacmd createNode -h server1.ibm.com -d /opt/IBM/ITM -u root
```

Important:

Unless you specifically indicate otherwise, the agent that you deploy using this command assumes that the monitoring server to which it connects is the monitoring server from which you run the command. The agent also uses the default settings for the communications protocol (IP.PIPE for protocol type and 1918 for the port). To change these defaults (especially if you are not using the IP.PIPE protocol), use the following property (specified with the -p parameter) when running the command: SERVER=[PROTOCOL://][HOST|IP][:PORT]. For example, SERVER=IP.PIPE://server1.ibm.com:1918.

Deploying non-OS agents

You can deploy non-OS agents through the Tivoli Enterprise Portal or from the command line.

- The deployment and configuration of agents varies depending on the specific agent. The following procedures provide generic deployment information. For the exact values required for your agent, see the configuration information in "Installing Response Time monitoring agents and related software" on page 101.
- Ensure that you have populated your agent depot, as described in "Populating the agent depot" on page 310, before attempting to deploy any agents.
- You must have already installed an OS agent on the computer where you are now deploying the non-OS agent and the agent must be running.

Deploying using the Tivoli Enterprise Portal

Before you deploy an agent using the Tivoli Enterprise Portal, application support for that agent must be installed on the portal server (see "Install application support for Linux and UNIX" on page 120 or "Install application support for Windows systems" on page 107).

Use the following steps to deploy an agent through the portal GUI:

1. Open the Tivoli Enterprise Portal.
2. In the Navigation tree, navigate to the computer to which you want to deploy the agent.
3. Right-click the computer and click **Add Managed System**.
4. In the **Select a Monitoring Agent** window, select the agent that you want to deploy and click **OK**.
5. In the **New Managed System Configuration** window, complete the configuration fields required for the agent. For information about these fields, see the configuration documentation for the agent that you are deploying.
6. Click **Finish**.
7. If the computer where you are deploying the agent already has a version of that agent installed, you can stop the deployment, add a new instance of the agent, if possible, or reconfigure the existing agent.

   A message will tell you when the deployment finishes successfully.

Deploying through the command line

To deploy non-OS agents from the command line, use the **tacmd addSystem** command. See the *IBM Tivoli Monitoring Command Reference* for the full syntax of this command, including parameter descriptions. You can run the **cinfo** command (UNIX) or the **kincinfo -i** command (Windows) to list the product codes for agents installed on the current computer.

For example, the following command deploys the Application Management Console, using the **CLI syntax tacmd addSystem** {-t|--type} *pc* {-n|--node} *MANAGED-OS* {-p|--property}**,** Where

-t|--type
specifies the type of agent to add to the monitoring system (see ITCAM for Transactions product codes).

-n|--node
Identifies the *node* or the directory on the monitoring system where the OS agent is installed, to which you want to add the agent. The name of a node includes the computer where the OS agent is installed and the product code for the OS agent. For example, stone.ibm.com:LZ is the name of the node on computer stone.ibm.com, which has a Linux OS agent installed.

-p|--property
Specifies SECTION.NAME=VALUE pairs that identify agent configuration properties and their values, where SECTION specifies the name of the section containing the key-value pair. KEY specifies the name of the configuration property, and VALUE specifies the property value. Refer to the agent configuration chapters for the agent that you want to deploy in "Installing Response Time monitoring agents and related software" on page 101 to obtain theKEY/VALUE pairs. See *IBM Tivoli Monitoring: Installation and Setup Guide* for complete details on using the property SECTION.NAME=VALUE pairs.

Each agent bundle has its own unique configuration parameters that you need to provide in this command. If you have an installed agent of the same type that you want to deploy, you can view the configuration parameters by running the following command from a monitoring server:

```
tacmd describeSystemType -t pc -p platform
```

An agent of the same type and platform must be deployed into the depot available to the monitoring server from which the command is run. For more information about agent-specific parameters, see the configuration chapters for the agent that you want to deploy in "Installing Response Time monitoring agents and related software" on page 101.

Upgrading non-OS agents remotely

You can upgrade non-OS agents through the Tivoli Enterprise Portal or from the command line.

- The deployment and configuration of agents varies depending on the specific agent. The following procedures provide generic deployment information. For the exact values required for your agent, see the configuration information in "Installing Response Time monitoring agents and related software" on page 101.
- Ensure that you have populated your agent depot, as described in "Populating the agent depot" on page 310, before attempting to deploy any agents.
- You must have already installed an OS agent on the computer where you are now deploying the non-OS agent and the agent must be running.

Upgrading through the command line

To upgrade non-OS agents from the command line, use the **tacmd updateagent** command.

See the *IBM Tivoli Monitoring Command Reference* for the full syntax of this command, including parameter descriptions.

You can run the **cinfo** command (UNIX) or the **kincinfo -i** command (Windows) to list the product codes for agents installed on the current computer.

Removing non-OS agents remotely

You can remove non-OS agents through the Tivoli Enterprise Portal or from the command line.

You can also uninstall non-OS agents from the Tivoli Enterprise Portal by stopping the agent and removing its configuration settings. After you have removed the agent from the Tivoli Enterprise Portal, you can completely uninstall the agent from the managed system. When you remove an agent, it is removed from any managed system lists to which it is assigned, any situation or policy distribution lists it was on, and any custom Navigator view items to which it was assigned.

Removing using the Tivoli Enterprise Portal

To remove a non-OS agent using the GUI:

1. Open the Tivoli Enterprise Portal.
2. In the Navigation tree, navigate to the computer from which you want to remove the agent.
3. Right-click the agent that you want to remove and select **Remove**.
4. Click **Yes** when you are asked to confirm the removal of the agent.
5. Click **Yes** when you are asked to confirm that you want to permanently uninstall the agent.

Removing through the command line

To remove non-OS agents from the command line, use the **tacmd removeSystem** command:

```
tacmd removeSystem -t product code -n Managed-OS
```

For example, to remove the Transaction Collector from `ibm001` which uses a Linux OS agent:

```
./tacmd removeSystem -t TU -n ibm001:LZ
```

See the *IBM Tivoli Monitoring Command Reference* for the full syntax of this command, including parameter descriptions.

You can run the **cinfo** command (UNIX) or the **kincinfo -i** command (Windows) to list the product codes for agents installed on the current computer.

Chapter 12. Configuring the Eclipse help server

You must configure the Tivoli Enterprise Portal client's Eclipse help server after installation is complete.

Before you begin

- Find out the port number for the Tivoli Enterprise Portal from the person who installed IBM Tivoli Monitoring
- If you are using an IBM Tivoli Monitoring version 6.1 with Fix Pack 3 on UNIX or Linux systems, the Eclipse server cannot start when it is installed. To resolve this problem, upgrade to ITM V6.2 FP1 iFix 1.

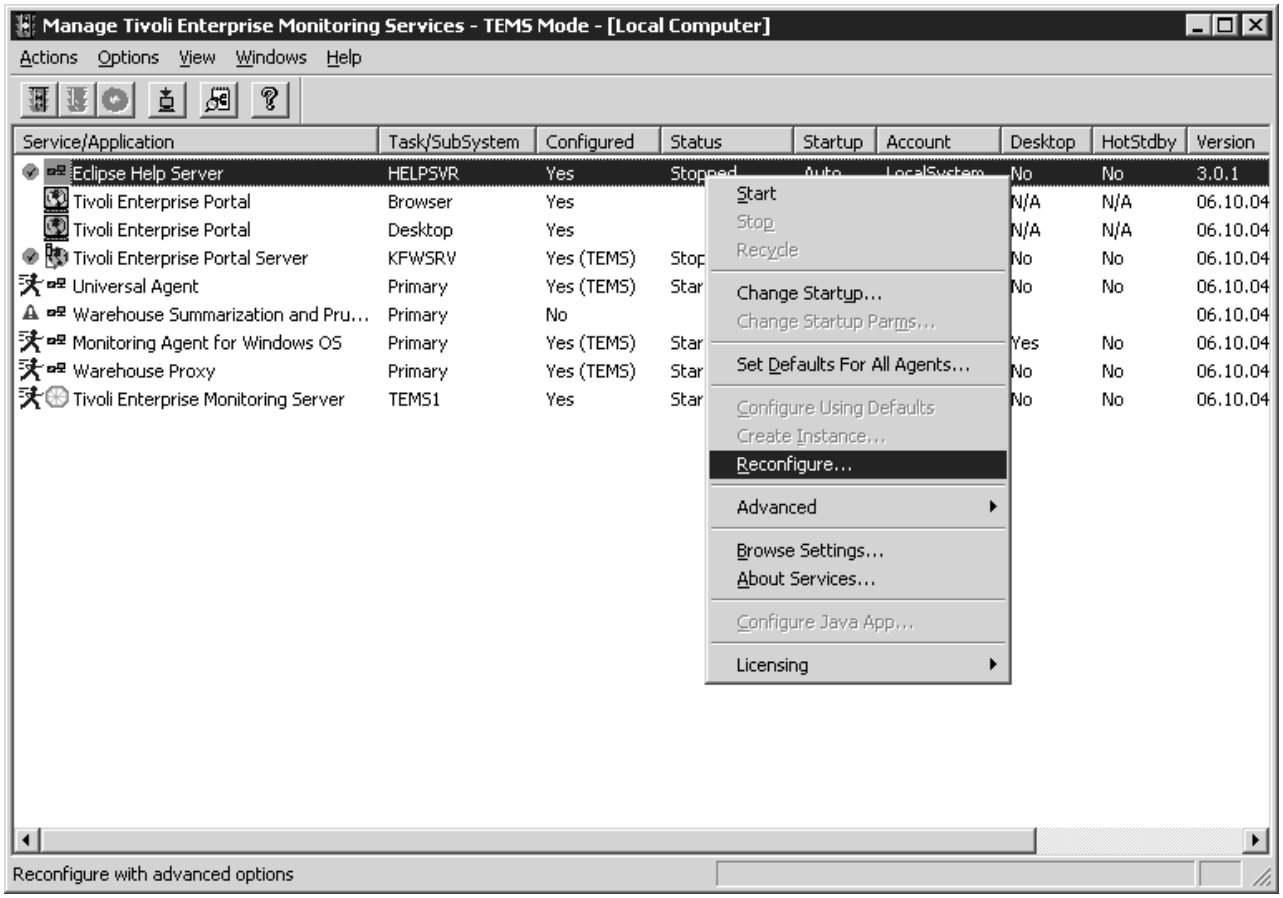
Procedure

1. Start Manage Tivoli Monitoring Services:

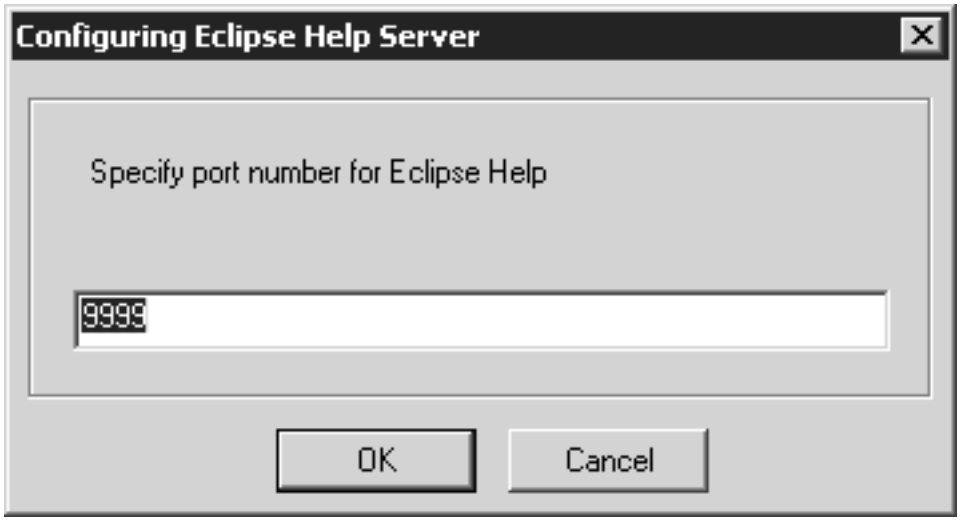
Windows	UNIX and Linux
Click **Start** > **All Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**	Run the following command: **./itmcmd manage**

   Note: The user interface might vary slightly, depending on the operating system.

2. Locate the Eclipse Help Server entry.
   a. If `Configured` displays `Yes`, go to step 4 on page 322.
   b. If `Configured` displays `No`, go to step 3 on page 322.



3. To configure the Eclipse Help Server:
   a. Right-click the Eclipse Help Server entry.
   b. Select **Configure Using Defaults** from the pop-up menu.



   c. Enter the port number specified when installing IBM Tivoli Monitoring.
   d. Click **OK**.
4. To set up the Eclipse help to automatically start the whenever this node is restarted,

a. Right-click the **Eclipse Help Server** entry.
b. Select **Change Startup** from the pop-up menu to display the Service Startup for Eclipse Help Server window.

c. Select **Automatic** at **Startup Type**.
d. Click **OK**.

Chapter 13. Starting and stopping servers and agents

Start and stop IBM Tivoli Monitoring components and Tivoli Enterprise Management Agents using the Manage Tivoli Enterprise Monitoring Services, or the command line.

Starting and stopping the Tivoli Enterprise Monitoring Server

Follow these directions to start or stop the Tivoli Enterprise Monitoring Server.

Start the server

On Windows platforms, perform the following steps:

1. Click **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
2. Right-click Tivoli Enterprise Monitoring Server.
3. Select **Start**.

On UNIX platforms, run the following command:

```
./itmcmd server start tems_name
```

Stop the server

On Windows platforms, perform the following steps:

1. Click **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
2. Right-click Tivoli Enterprise Monitoring Server.
3. Select **Stop**.

On UNIX platforms, run the following command:

```
./itmcmd server stop tems_name
```

tems_name is the name of the monitoring server

Starting and stopping the Tivoli Enterprise Portal server

Follow these directions to start or stop the Tivoli Enterprise Portal server:

Start the portal server

On Windows platforms, perform the following steps:

1. Click **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
2. Right-click Tivoli Enterprise Portal Server.
3. Select **Start**.

On UNIX platforms, run the following command:

```
./itmcmd agent start cq
```

Stop the portal server

On Windows platforms, perform the following steps:

1. Click **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
2. Right-click Tivoli Enterprise Portal Server.
3. Select **Stop**.

On UNIX platforms, run the following command:

```
./itmcmd agent stop cq
```

Starting and stopping the Tivoli Enterprise Portal desktop client

Follow these directions to start or stop the Tivoli Enterprise Portal desktop client

Start the desktop client

On Windows platforms, perform the following steps:

1. Click **Start** > **Programs** > **IBM Tivoli Monitoring** > **Tivoli Enterprise Portal desktop**.
2. Enter your user ID and password on the logon window. The default user ID is `sysadmin`.
3. Click **OK**.

On UNIX platforms, run the following command:`./itmcmd agent start cj`

Stop the desktop client

On Windows platforms, perform the following steps:

1. Click **Start** > **Programs** > **IBM Tivoli Monitoring** > **Manage Tivoli Monitoring Services**.
2. Right-click Tivoli Enterprise Portal desktop.
3. Select **Stop**.

On UNIX platforms, run the following command:`./itmcmd agent start cj`

Starting and stopping monitoring agents

Start and stop monitoring agents using the Manage Tivoli Enterprise Monitoring Services or the from the user interface in both Windows and UNIX environments. You can also use the command line in UNIX.

You can also run the `ITMAgents1` script from the `/etc/init.d` directory. The location varies for different environments. Only run the `ITMAgents1` script if the system restarts other Response Time agents.

Alternatively, you can restart the Transaction Tracking agents from the Tivoli Enterprise Portal if required by using the Proxy Agent Services. See the IBM Tivoli Monitoring Information Center for further information.

If an OS agent is installed on the same computer as an agent and shares the same `CANDLE_HOME` directory, you can remotely stop and start the agent from the Tivoli Enterprise Portal. Right click on the agent in the Tivoli Enterprise Portal and select **Stop** or **Restart**.

Note: Your user ID must have the proper permission to start and stop agents. See the IBM Tivoli Monitoring documentation for more information about permissions.

Note: If you install Response Time on SUSE SLES 10 platform, the agent might not restart automatically when the environment reboots. You can start the agent manually with instructions in this section.

Follow these directions to start or stop the monitoring agents:

Start a monitoring agent

On Windows platforms, perform the following steps:

1. Access the Navigator.

2. Right-click the monitoring agent that you want to start
3. Select  .

   On UNIX platforms, run the following command:

   `./itmcmd agent start pc`

Stop a monitoring agent

On Windows platforms, perform the following steps:

1. Access the Navigator.
2. Right-click the monitoring agent that you want to stop.
3. Select  .

   On UNIX platforms, run the following command:

   `./itmcmd agent stop pc`

Where pc is the product code for the monitoring agent that you want to start or stop. See Table 11 on page 48 for a list of product codes.

For example to start Tivoli Enterprise Portal desktop client, run the following command:`./itmcmd agent start t5`

Note: On UNIX systems, the Web Response Time monitoring agent occasionally gives the following error: `[root@rh5ma bin]# ./CandleAgent -h /opt/IBM/ITM -c stop t5 Stopping ITCAM for Web Response Time ... Product t5 was not stopped.` If this happens, use `-f` to force Web Response Time to stop. If you are doing a remote uninstallation for Web Response Time, you should also use this option to stop the monitoring agent *before* doing the remote uninstall.

Appendix A. Installing and uninstalling the language pack

By default, ITCAM for Transactions is enabled for the English language. If you want to use the product in another language, you must install the translated language pack. The language packs are available on the language support image for each component.

ITCAM for Transactions provides language packs in the following languages:

- Brazilian
- French
- German
- Hungarian
- Italian
- Japanese
- Korean
- Polish
- Portuguese
- Russian
- Simplified Chinese
- Spanish
- Traditional Chinese

The National Language Support resources for ITCAM for Transactions agent are provided on a separate CD that is shipped with ITCAM for Transactions.

Note: Before you can install a language pack, you must install the components of ITCAM for Transactions in English.

Installing and uninstalling a language pack on Windows systems

Before you begin installing the language pack, make sure that you comply with the following requirements:

- Install the English version of ITCAM for Transactions.
- Determine the installation directory of ITCAM for Transactions on the computer where you plan to run the language pack installation program. You must install the language packs in the same directory.
- The default installation directory for the Tivoli Enterprise Portal Server and the Tivoli Enterprise Monitoring Agent is `C:\IBM\ITM`.
- The Language Pack installation requires Java Runtime Environment (JRE) 1.4.x or above. Use the JRE that was installed by IBM Tivoli Monitoring (on the Tivoli Enterprise Portal) or the JRE installed by the base product driver. The default path for that JRE is `C:\Program Files\IBM\Java142`.

Use one of the following methods for installing or uninstalling a language pack on Windows:

- "Installing a language pack on Windows systems" on page 330
- "Silently installing a language pack on Windows" on page 330
- "Uninstalling a language pack on Windows systems" on page 331

Install separate language packs for each of the ITCAM for Transactions subcomponents:

- Internet Service Monitoring
- Response Time
- Transaction Tracking

Installing a language pack on Windows systems

The language pack installation programs are provided on the *ITCAM for Transactions Language Support* CD.

The language pack must be installed on the computer running the Tivoli Enterprise Portal and the computer running the agent.

Note: If the agent is installed on the same computer as the Tivoli Enterprise Monitoring Server, install the language pack on both the Tivoli Enterprise Portal Server and the Tivoli Enterprise Monitoring Server.

To install the language pack:

1. Run `lpinstaller.exe` from the Language Pack CD.
2. Select the language of the installer and click **OK**.
3. Click **Next** on the Introduction window.
4. Click **Add/Update** and then click **Next**.
5. Select the folder in which the National Language Support package (`NLSPackage`) files are located and click **Next**.
6. Select the language support for the required agent and click **Next**.

   Tip: Hold down the Ctrl key to select multiple entries.
7. Select the languages that you want to install and click **Next**.
8. Verify the installation summary page and click **Next** to begin the installation.
9. Click **Next**.
10. When the installation is complete, click **Finish** to exit the installer.
11. Restart the agent and any relevant IBM Tivoli Monitoring components installed on the computer.

Silently installing a language pack on Windows

To install the language pack using the silent installation procedure, use a response file. The response file enables you to run the installation in silent mode without user interaction. The ITCAM for Transactions language pack provides you with the `ITM_Agent_LP_silent.rsp` template response file that you can customize to reflect the correct installation directories.

To run a silent installation, perform the following steps:

1. Copy `ITM_Agent_LP_silent.rsp` to the directory where `lpinstaller.sh` is located (that is, the IBM Tivoli Monitoring agent language pack build location.)
2. Modify the response file so that it is correct for your environment. For example, set the following parameters in the response file:

   ```
   INSTALLER_UI;
   CHOSEN_INSTALL_SET;
   NLS_PACKAGE_FOLDER;
   PROD_SELECTION_PKG;
   ```

BASE_AGENT_FOUND_PKG_LIST;
LANG_SELECTION_LIST

3. From a command line run the following command:
   `lpinstaller.bat -f responseFileName`

   Where, *responseFileName* is the fully qualified path to the response file (either the `ITM_Agent_LP_silent.rsp` default file or one that you customize) containing the installation options.

   For example, `lpinstaller.bat –f ITM_Agent_LP_silent.rsp`.

Uninstalling a language pack on Windows systems

Before uninstalling a language pack, ensure that ITCAM for Transactions, IBM Tivoli Monitoring, the Java runtime environment and the language pack are still installed.

Follow these steps to remove the language pack:

1. Run `lpinstaller.exe` from the Language Pack CD.
2. Select the language of the installer and click **OK**.
3. Click **Next** on the Introduction window.
4. Click **Remove** and then click **Next**.
5. Select the language support for the required agent, and click **Next**.

   Tip: Hold down the Ctrl key to select multiple entries.
6. Select the languages that you want to uninstall and click **Next**.
7. Verify the installation summary page and click **Next** to begin uninstalling.
8. When the process is complete, click **Finish** to exit the installer.
9. Restart the agent and any relevant IBM Tivoli Monitoring components installed on the computer.

Installing and uninstalling a language pack on Linux or UNIX systems

Before you begin installing the language pack, complete the following tasks:

- Install the English version of ITCAM for Transactions.
- Verify the installation directory of ITCAM for Transactions on the computer where you plan to run the language pack installation program. You must install the language packs to the same directory.
- Verify that the default installation directory for the Tivoli Enterprise Portal and the Tivoli Enterprise Monitoring Agent is: `/opt/IBM/ITM.`
- The Language Pack installation requires Java Runtime Environment (JRE) 1.4.x or above. Use the JRE that was installed by IBM Tivoli Monitoring (on Tivoli Enterprise Portal Server) or the JRE installed by the base product driver. The default path for that JRE is `/opt/IBM/ITM/JRE.`

Choose one of the following methods for installing or uninstalling a language pack on Linux or UNIX:

- "Installing a language pack on Linux or UNIX systems" on page 332
- "Silently Installing a language pack on Linux or UNIX" on page 333
- "Uninstalling a language pack on Linux and UNIX systems" on page 333

Install separate language packs for each of the ITCAM for Transactions subcomponents:

- Internet Service Monitoring
- Response Time
- Transaction Tracking

Installing a language pack on Linux or UNIX systems

The language pack installation programs are provided on the *ITCAM for Transactions, Language Support CD*.

The language pack must be installed on both the computer running the Tivoli Enterprise Portal and the computer running the agent.

Note: If the agent is installed on the same computer as the Tivoli Enterprise Monitoring Server, install the language pack on both the Tivoli Enterprise Portal Server and the Tivoli Enterprise Monitoring Server.

To install the language pack:

1. Run the following command to create a temporary directory on the computer. Make sure that the full path of the directory does not contain any spaces:
   `mkdir dir_name`
2. Mount the language pack CD to the temporary directory you just created.
3. Run the following command to start the installation program:
   ```
   cd dir_name
   lpinstaller.sh -c ITM Home Directory [-i install_mode]
   ```
   Where:
   ITM Home Directory is where you installed IBM Tivoli Monitoring, usually `/opt/IBM/ITM` for AIX and Linux systems.
   install_mode is either `gui`, `console`, or `silent`. For Turkish, use `GUI` instead of`gui`.
4. Select the language of the installer and click **OK**.
5. Click **Next** on the Introduction window.
6. Click **Add/Update** and then click **Next**.
7. Select the folder in which the National Language Support package (`NLSPackage`) files are located and click **Next**.
8. Select the language support for the required agent and click **Next**.

   Tip: Hold down the Ctrl key to select multiple entries.
9. Select the languages that you want to install and click **Next**.
10. Verify the installation summary page and click **Next** to begin the installation.
11. Click **Next**.
12. When the installation is complete, click **Finish** to exit the installer.
13. Restart the agent and any relevant IBM Tivoli Monitoring components installed on the computer.

Silently Installing a language pack on Linux or UNIX

To install the language pack using the silent installation procedure you use a response file. The response file enables you to run the installation in silent mode without user interaction. The ITCAM for Transactions language pack provides you with the `ITM_Agent_LP_silent.rsp` template response files. This is the template response file that you can customize to reflect your correct installation directories.

Note: Before installing a language pack using silent mode, ensure that you have already installed the product in English.

To run a silent installation, perform the following steps:

1. Copy `ITM_Agent_LP_silent.rsp` to the directory where `lpinstaller.sh` is located (that is, the IBM Tivoli Monitoring agent language pack build location.)
2. Modify the response file so that it is correct for your environment. For example, set the following parameters in the response file:

   INSTALLER_UI;

   CHOSEN_INSTALL_SET;

   NLS_PACKAGE_FOLDER;

   PROD_SELECTION_PKG;

   BASE_AGENT_FOUND_PKG_LIST;

   LANG_SELECTION_LIST
3. Run the following command to create a temporary directory, ensuring that the full directory path does not contain any spaces:

   `mkdir dir_name`
4. Mount the language pack CD to the temporary directory that you just created.
5. Run the following command to launch the installation program:

   ```
   cd dir_name
    lpinstaller.bat -c ITM Home directory -i silent -f responseFileName
   ```

   Where:

   - *ITM Home directory* is the directory to which IBM Tivoli Monitoring is installed. For example, `/opt/IBM/ITM` for AIX and Linux.
   - *responseFileName* is the fully qualified path to the response file (either the default `ITM_Agent_LP_silent.rsp` file or a file that you customize) containing the installation options.
6. Restart the agent and any relevant IBM Tivoli Monitoring components installed on the computer.

Uninstalling a language pack on Linux and UNIX systems

Before uninstalling a language pack, ensure that ITCAM for Transactions, IBM Tivoli Monitoring, the Java runtime environment and the language pack are still installed.

Follow these steps to remove the language pack:

1. Run the following command to create a temporary directory on the computer. Make sure that the full path of the directory does not contain any spaces:
   `mkdir dir_name`
2. Mount the language pack CD to the temporary directory you just created.
3. Run the following command to start the installation program:

   ```
   cd dir_name
   ./lpinstaller.sh -c ITM Home Directory [-i install_mode]
   ```

Where:

ITM Home Directory is where you installed IBM Tivoli Monitoring, usually `/opt/IBM/ITM` for AIX and Linux systems.

install_mode is either `gui`, `console`, or `silent`. For Turkish, use `GUI` instead of`gui`.

4. Select the language of the installer and click **OK**.
5. Click **Next** on the Introduction window.
6. Click **Remove** and then click **Next**.
7. Select the language support you want to uninstall and click **Next**.

   Tip: Hold down the Ctrl key to select multiple entries.
8. Select the languages that you want to uninstall and click **Next**.
9. Verify the installation summary page and click **Next** to begin uninstalling.
10. When the process is complete, click **Finish** to exit the installer.
11. Restart the agent and any relevant IBM Tivoli Monitoring components installed on the computer.

Appendix B. Internet Service Monitoring open ports

Internet Service Monitoring opens ports on the host computer when it is installed.

Table 64 lists the ports opened on the host machine.

Table 64. Default ports for Internet Service Monitoring

Connection	Default Port	Configuration
Databridge to Internet service monitoring agent	`9520/tcp`	To change the default port, edit the `TEMAPort` property in the `kisagent.props` file. See Table 15 on page 93 for further information.
Service monitors to Databridge	`9510/tcp`	To change the default port, edit the `SocketPort` property in the `bridge.props` file. See "Databridge properties and command line options" on page 88 for further information.

Appendix C. Internet Service Monitoring directory structure

Internet Service Monitoring installs to a subdirectory of the ITM directory structure, known as ISMHOME. On Windows systems, ISMHOME is c:\IBM\ITM\tmaitm6\ism; on Linux and UNIX systems it is /opt/IBM/ITM/*arch*/is.

Table 65 lists the directory structure for Internet Service Monitoring version V7.2 and explains the purpose of each directory.

Table 65. Internet service monitor directory structure

Directory	Purpose
$ISMHOME/certificates	Directory for the Databridge SSL certificate files.
$ISMHOME/datalogs	Root directory for datalog files that are generated by the Datalog module.
$ISMHOME/datalogs/default	Directory for the default datalog files that are used by the Datalog module.
$ISMHOME/etc/props	Directory for system properties files. Properties file control the operation of the system components including the Internet service monitors.
$ISMHOME/etc/rules	Directory for rules files that are used by the ObjectServer module and the Internet service monitors to convert events into Netcool/OMNIbus alerts.
$ISMHOME/log	Directory for system log files as well as error files (.err).
$ISMHOME/mibs	Directory for MIB files that are used by SNMP-based Internet service monitors.
$ISMHOME/objectserver/bin	Directory for the interfaces file that is used to connect the ObjectServer module to an ObjectServer on UNIX systems.
$ISMHOME/objectserver/etc	Directory for the connections data file (omni.dat) that is used to connect the ObjectServer module to an ObjectServer on UNIX systems.
$ISMHOME/platform/*arch*/bin	Directory for system executable files and the command line configuration utility (ismbatch). *Arch* is the name of the platform. For example, Win32 for Windows systems.
$ISMHOME/profiles	Root directory used for profiles.
$ISMHOME/profiles/active	Directory used for active profiles.
$ISMHOME/profiles/default	Directory used for default profile templates.
$ISMHOME/profiles/deleted	Directory used for deleted profiles and profiles that are corrupt.
$ISMHOME/profiles/inactive	Directory used for inactive profiles.
$ISMHOME/profiles/obsolete	Directory used for profiles that are no longer compliant with the current version of Internet Service Monitoring .

Table 65. Internet service monitor directory structure (continued)

Directory	Purpose
`$ISMHOME/scripts`	Directory used for the Internet Service Monitoring script files.
`$ISMHOME/scripts/SAA`	Directory used for the SAA monitor script files.
`$ISMHOME/scripts/RPING`	Directory used for the RPING monitor script files.
`$ISMHOME/var`	Directory for the Raw Capture file that is used by the ObjectServer, also the directory for the Store and Forward file that is used by the Databridge.
`%ISMHOME%\objectserver\ini`	Directory for the connections data file (`sql.ini`) that is used to connect the ObjectServer module to an ObjectServer on Windows systems.
`%CANDLE_HOME%\TMAITM6`	Directory for the connection information that connects the Internet service monitoring agent to the IBM Tivoli Monitoring module on Windows systems.

Appendix D. Upgrading from previous versions of Response Time to V7.3

Describes how to upgrade your Response Time monitoring environment.

This chapter provides the information you need to collect before you begin an upgrade, the steps involved in the upgrade, and any follow-up tasks that you might need to perform.

Note: After you complete the upgrade, data from previous versions of the Response Time agent is available only in the Tivoli Data Warehouse. You can see this information by looking at BIRT reports (described in the *IBM Tivoli Composite Application Manager for Transactions Administrator's Guide*) or from the ITCAM for End User Response Time Dashboard, version 6.2.

Before you begin

This section provides information about decisions and actions you should take before you begin an upgrade.

- Check the prerequisites information about the required hardware and software. See Prerequisites in the ITCAM for Transactions Information Center.
- Do not upgrade the Java version on a system running Response Time 6.2 Robotic or Dashboard agents to Java 1.5 until you upgrade those agents to version 7.1 or later. If you do, you cannot start the monitoring agents.
- For any custom situations that you created from product-provided situations, you must make a copy of the product provided situation and give it a different name. All product provided situations are reset after an upgrade and you lose any customized situations that still have the name of the product provided situations.
- Know the product codes for the monitoring agent that you are installing. See Table 11 on page 48.
- Collect the information you need to know before you begin the upgrade. See Information to collect before you begin installation and configuration.
- You cannot have different versions of either Client Response Time (t4) or Robotic Response Time (t6) installed on the same computer, because the agents will not correctly generate data. For example, you cannot have Client Response Time 6.2 FP1 and Client Response Time V7.3 on the same computer.
- If you are upgrading to the Application Management Console monitoring agent, copy the robotic scripts and the Rational Performance Tester runtime from the version 6.2 environment to the current environment.

  Copy the robotic scripts located at `<ITM>/kt1depot/T3`, where `<ITM>` is the directory where IBM Tivoli Monitoring is installed, into the `<ITM>/kt1depot/T3/<type>` folder, where `<type>` is one of the following directories, based on the type of robotic script:
  - **Command Line**: `CLI_PLAYBACK`
  - **LoadRunner**: `LOADRUNNER`
  - **Robot GUI**: `ROBOT_GUI`
  - **Rational Performance Tester**: `RPT`
  - **Robot VU**: `ROBOT_VU`

- Install theApplication Management Console agent before you install any other Response Time agent.
- If the scripts are copied directly from a previous version of the Application Management Console agent to the V7.3 agent, the scripts will be missing the subtransaction information when viewed in the Application Management Configuration Editor. The subtransaction information is added to scripts in V7.3 and the scripts must be uploaded again from the Rational Performance Tester workbench using the export wizard. When these scripts are uploaded again, meta-information about the subtransactions is included in the exported script. This subtransaction meta-information is only applicable to Rational Performance Tester scripts. All other scripts can be directly copied from a previous Application Management Console agent.

Process overview of upgrade

Upgrading a monitoring agent involves the *same* steps as installing an agent. This section provides an overview of the upgrade process with references to sections in this document where you can find additional information:

Table 66. Task overview of upgrade

What to do	Where to find more information
Obtain the installation software. **Note:** There is not a separate upgrade image; it is included in the regular installation software.	Either download the software from Passport Advantage (see the Download Document in the ITCAM for Transactions Information Center) or use a product CD. Talk to your Tivoli System Administrator.
Install the latest versions of Rational products: • Rational Performance Tester *(Required for Robotic Response Time)* Rational Performance Tester is only required on systems where the scripts are recorded and developed, but is not required on systems where the agents are located. The export wizard must be at the latest version to support the latest versions of Rational Performance Tester. If you intend to reuse an existing installation of Rational Performance Tester version 8.0 or later, run the Integration Support installer to update the export wizard to the latest version. The Integration Support installer also applies additional hotfixes that are mandatory in ITCAM environments. • Rational Robot *(Optional)* Rational Robot is also required if you intend to deploy Rational Robot scripts on Robotic Response Time agents.	"Installing integration support for Rational Performance Tester" on page 130 "Installing Rational Robot" on page 135
Install the Tivoli Enterprise Monitoring Agent on a computer by itself without Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal. Agent specific application support on these components is installed on a different computer.	• "Install a monitoring agent on Windows systems" on page 102 • "Installing a monitoring agent on Linux or UNIX computers" on page 116 • "Performing a silent installation" on page 126

Table 66. Task overview of upgrade (continued)

What to do	Where to find more information
Install Tivoli Enterprise Monitoring Agent specific application support on a computer other than where the monitoring agent is installed.	• "Install application support for Windows systems" on page 107 • "Install application support for Linux and UNIX" on page 120
Enable file transfer to ITCAM for Transaction agents.	"Enabling File Transfer (UNIX and Linux)" on page 122
Configure the monitoring agent.	• "Configuring Application Management Console" on page 143 • "Configuring Client Response Time" on page 149 • "Configuring Robotic Response Time" on page 154 • "Configuring Web Response Time" on page 164
Verify the installation of a Tivoli Enterprise Monitoring Agent.	"Verify installations of Response Time monitoring agents" on page 125
Configure the Eclipse help server.	Chapter 12, "Configuring the Eclipse help server," on page 321
Setup remote deployment. *(Optional)* Response Time supports remotely deploying the monitoring agent across your environment from a central location, the Tivoli Enterprise Monitoring Server.	Chapter 11, "Working remotely," on page 309
Configure IBM Tivoli Monitoring to forward events to Tivoli Enterprise Console. *(Optional)*	Appendix E, "Tivoli Enterprise Console event mapping," on page 349
Reboot the computer on which you installed and configured the agent.	
Start the monitoring agent.	
Configure for historical data collection. *(Optional)*	Setting up historical data collection with Response Time in the *Administrator's Guide*
Install a language pack. *(Optional)* You can install local language support on each computer on which the Tivoli Enterprise Portal is located	Appendix A, "Installing and uninstalling the language pack," on page 329

Follow-up or related tasks

The following are tasks for after you finish the upgrade:

Upgrading situations to profiles

You can run a command-line utility to migrate Response Time 6.2, Fix Pack 1, configuration situations to Response Time V7.3 client, transaction and profile XML files. See "Upgrading situations to profiles automatically" on page 343.

Historical data collection

If you enabled historical data collection, you should disable warehouse data collection for the *over time* tables after completing an upgrade.

Web Response Time workspaces are empty after an upgrade.

All workspaces are blank until new data collection occurs. The V7.3 workspaces use a different set of queries and tables than 6.2.

After upgrading to the Application Management Console (t3), some previous clients and servers appear as gray entries in the Tivoli Enterprise Portal Navigator.

This is because they are no longer displayed in this location; they are now displayed under each specific Application under the Clients and Servers sub-nodes. You must clean up any offline entries in the TEP by doing the following:

1. Right-click **Enterprise → Workspaces → Managed System Status**.
2. Select all **OFFLINE** entries.
3. Right-click and select **Clear Offline Entry**.

If the monitoring agents fail to start immediately after an upgrade from 6.2.1 to V7.3 *and*

you receive error messages similar to the following one in the agent logs :

```
(486AA0F5.0000-E78:kt2sjni.cpp,1065, "SimpleJNI::runInitialization")
Error: Unable to find AgentJNI
(486AA0F5.0001-E78:kt2sjni.cpp,523,"SimpleJNI::loadVM") Exiting ...
```

Agent logs are located in `<ITM>/tmaitm6/logs/` (for Windows) or `<ITM>/logs` for (UNIX).

`<ITM>` is the directory where IBM Tivoli Monitoring is installed.

If this occurs, do the following:

1. Configure the agent again. See the following chapters:
   - "Configuring Client Response Time" on page 149
   - "Configuring Robotic Response Time" on page 154
   - "Configuring Web Response Time" on page 164
2. Restart the agent. See Starting or stopping monitoring agents in the *Administrator's Guide*.

If you install the RT 7.1 Application Management Console on Red Hat Enterprise Linux AS release 4 and run the agent configuration command `itmcmd config -A t3`

The following error is written to `stdout`: `KCIIN0231E Could not update file : java.lang.IllegalArgumentException: Unknown section name: ttdbconfig`. The following exception is written to `itm_config.trc`:

```
2008-07-01 15:57:36.274-06:00 com.ibm.tivoli.monitoring.install.cmdline.AgentNode getSectionSchema
__ERROR__ (traceKey:1214945856274)
java.lang.IllegalArgumentException: Unknown section name: ttdbconfig
        at com.ibm.tivoli.monitoring.install.cmdline.AgentNode.getSectionSchema(AgentNode.java:109)
        at com.ibm.tivoli.monitoring.install.cmdline.SectionNode.&lt;init>(SectionNode.java:45)
        at com.ibm.tivoli.monitoring.install.cmdline.InstanceNode.&lt;init>(InstanceNode.java:67)
        at com.ibm.tivoli.monitoring.install.cmdline.AgentNode.AgentNodeConstructor(AgentNode.java:79)
        at com.ibm.tivoli.monitoring.install.cmdline.AgentNode.&lt;init>(AgentNode.java:61)
        at com.ibm.tivoli.monitoring.install.cmdline.KinCmdLineConfig.KinCmdLineConfigConstructor(KinCmdLineConfig.java:141)
        at com.ibm.tivoli.monitoring.install.cmdline.KinCmdLineConfig.&lt;init>(KinCmdLineConfig.java:91)
        at com.ibm.tivoli.monitoring.install.cmdline.KinCmdLineConfig.runConsole(KinCmdLineConfig.java:379)
        at com.ibm.tivoli.monitoring.install.KinPlugin.ttyConfig(KinPlugin.java:75)
        at ITMinstall.CandleConfigAgent.main(CandleConfigAgent.java:601)
```

Do the following tasks:

1. Delete the log file, *agent*`_t3.log`, located in the *ITM*`/config` directory on UNIX systems, and *ITM*`/TMAITM6` on Windows systems.

   ITM is the directory where IBM Tivoli Monitoring is installed.
2. Run the agent configuration again. See "Configuring Application Management Console from the GUI (UNIX and Linux)" on page 146

Summarization behavior for the RRT_Playback_Status table has changed to be more accurate

If you collect Summarization (S&P) data for the RRT_Playback_Status tables, do the following after upgrading your Tivoli Enterprise Monitoring Server andTivoli Enterprise Portal support for Robotic Response Time.

Note: This procedure applies only if you summarize PLAYBACK status.

1. Upgrade the Robotic Response Time Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal support for yourIBM Tivoli Monitoring environment.
2. Stop the Warehouse Proxy, the Summarization and Pruning Agent, theTivoli Enterprise Monitoring Server, and the Tivoli Enterprise Portal.
3. Inside the Tivoli Data Warehouse database, drop the RRT_Playback_Status Summary tables and views. You do not have to drop the original source table. These tables have names like RRT_Playback_Status_H. For example to drop the hourly summary table issue the following command:

```
drop table "RRT_Playback_Status_H";
drop view "RRT_Playback_Status_HV";
commit;
```

4. Restart the Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal, and Warehouse Proxy and Summarization and Pruning Agent.

The Summary tables are recreated the next time the Summarization and Pruning agent runs.

Re-configuring and restarting agents that are not upgraded

If after upgrading from a previous version you still have Response Time agents in your environment for a previous version that were not upgraded, be sure to manually re-configure and restart those agents to operate correctly.

Upgrading situations to profiles automatically

You can run a command-line utility to upgrade Response Time 6.2, Fix Pack 1, configuration situations to Response Time 7.1 or later client, transaction and profile XML files. This upgrade includes Client Response Time, Robotic Response Time, and Web Response Time agents. The utility upgrades and converts all situations into as few profiles as possible. For example, the tool merges situations with the same agent type and distribution lists into a single profile.

Previously, you needed to complete this upgrade process manually. For information on this manual process, see the previous version of this guide.

What is not upgraded

The following items are not upgraded:

- Situations that define realms or thresholds.
- Existing client groups, transactions, and profiles from the Application Management Console (T3) depot. These items are not overwritten. Instead, the utility prints a message to the console indicating that it found a duplicate object. You can force an overwrite of duplicated data with the -f command line option.
- Configuration settings in the **Expert Advice**, **Action**, and **Until** tabs in the Situation Editor.
- Situations that are not set to run at startup.

All messages are in English.

Before you begin

Do the following items before you begin the upgrade:

- Upgrade the monitoring agents and agent support to version 7.2. See "Installing Response Time monitoring agents and related software" on page 101.
- Upgrade Application Management Console (T3) agent and support on both the computer running the agent and on the Tivoli Enterprise Portal. See "Installing Response Time monitoring agents and related software" on page 101.

IBM Tivoli Monitoring authorization requirements

The IBM Tivoli Monitoring permissions required to run this tool are the same as those required to display Response Time situations in the Situation Editor. Specifically, a user running the tool needs to have the following permissions:

- A permission setting of **View** or **Modify** on the Situation authority.
- The **Allowed Applications** must include all of the following installed agent types (Robotic Response Time, Web Response Time, Client Response Time).

If a user cannot access the Situation Editor, or the Situation Editor does not show situations for the expected agent types, verify that these permissions are granted.

Running the upgrade utility

Follow these steps to run the utility on the Tivoli Enterprise Portal Server:

1. Access the command line for the system on which the utility is installed.
   Locate the `sit2profile` upgrade utility in one of the following directories:
   - **Windows:** `C:\IBM\ITM\CNB\classes\sit2profile.cmd`
   - **UNIX/Linux:** `/opt/IBM/ITM/<platform>/cw/classes/sit2profile.sh`
2. Run the following command:
   ```
   sit2profile.[cmd|sh] –u <username> [-s] [ -p <password>| -i ] [-f]
   ```
   The following options are specified in this syntax:

   -u The TEP server user name, such as `sysadmin`.

   -s Ignore the situation *Run at startup* option, migrate all situations.

   -p An optional command line password for the specified server user name.

   -i If a password is not specified with the `-p` option, this option prompts the user for the password on standard input (`stdin`).

   -f Forces existing client groups, transactions or profiles to be overwritten.

Error Codes

The command-line error codes returned from the utility are listed in Table 67.

Table 67. Command line error return codes

Error code	Meaning
0	Success.
1	Incorrect or missing CLI argument.

Table 67. Command line error return codes (continued)

Error code	Meaning
2	Unable to contact the Tivoli Enterprise Portal Server.
3	Unable to contact an Application Management Console (T3) agent.
4	Invalid user ID or password.
5	An internal error has occurred.

Mapping attributes from situations to clients, transactions, and profiles

When the utility discovers a situation that contains a command in the Action tab, it prints a generic message stating that actions are not upgraded.

Client groups

When client situations are converted into client groups, it is a one-to-one mapping (one new client group for each client situation). The upgraded client situation attribute groups are `WRT_Client_Patterns`, `CRT_Client_Patterns`, and `RRT_Client_Patterns`.

Table 68. Client situations to client groups mapping

Situation Attribute	Application Management Configuration Editor property
Situation Name	Client Group Name
Situation Description	Client Group Description
Client Name	Client Reporting Rule. For more information, see the table, Table 69 and also see the table, Table 70
Client Hostname Pattern	Hostname `include` filter value
Client IP	IP `include` filter value
Aggregates Uniquely (t6 and t4 situations)	See the table, Table 69
Aggregation (t5 situations only)	See the table, Table 70
Situation Agent Type	Agent Type `include` filter value, such as **T4** for Client Response Time, **T5** for Web Response Time, or **T6** for Robotic Response Time.

Table 69. Aggregates Uniquely situation column to client reporting value mapping (Client Response Time and Robotic Response Time

Aggregates Uniquely Column	Client Reporting Value
TRUE	`$Hostname$`
FALSE	Situation Client Name

Table 70. Aggregation situation column to client reporting value mapping (Robotic Response Time only)

Aggregation Column	Client Reporting Value
Aggregate Pattern	Situation Client Name
Aggregate Subnet	`$IPV4ClassCSubNet$`
Aggregate Uniquely	`$IP$`

Robotic transactions

Non-CLI robotic transactions (such as Rational Performance Tester, Robot GUI, Robot VU, or Mercury LoadRunner) are created automatically from the recording files stored on the Application Management Console agent and should already be in the Applications view, so no upgrade is required.

Non-robotic transactions

Non-Robotic transactions are upgraded from situation tables CRT_Transaction_Patterns and WRT_Transaction_Patterns using the mapping in this table. There is a one-to-one mapping from each row in a situation formula to a transaction.

Table 71. Non-robotic transactions to Application Management Configuration Editor Robotic transaction mapping

Situation Attribute transaction property	Application Management Configuration Editor Transaction Property
Situation Name	Transaction Name
Situation Description	Transaction Description
Application Name	Application Name, Application Reporting Rule Table 72
Application Pattern	Application Name `include` filter value
Aggregate Applications Uniquely	See Table 72
Transaction Name	Transaction Reporting Rule (see Table 73)
Transaction Pattern	Transaction Name `include` filter value
Aggregate Transactions Uniquely	See Table 73

Table 72. Aggregates Applications Uniquely situation column to transaction reporting value mapping

Aggregate Applications Uniquely Column	Transaction Reporting Value
TRUE	`$ApplicationName$` for Client Response Time `$ContextRoot$` for Web Response Time
FALSE	Situation Application Name

Table 73. Aggregates Transactions Uniquely situation column to transaction reporting value mapping

Aggregate Applications Uniquely Column	Transaction Reporting Value
TRUE	`$TransactionName$` for Client Response Time `$URLPath$/$URLFile$` for Web Response Time
FALSE	Situation Transaction Name

Profiles

Situations are converted into as few profiles as possible, where all situations of each agent type with the same distribution list are grouped into a single profile. The algorithm used when locating a profile is as follows:

1. Search for an existing profile of the same agent type that has a matching distribution list (e.g. the exact same agents and managed system lists).

2. If a matching profile is found, then assign the transaction to this profile. If a transaction of the same name already exists in the profile, then it will not be overwritten unless the `-f` option is given on the command line.
3. If a matching profile is not found, then a new profile is create using the situation name as the profile name. If a profile of this agent type already exists under the situation name, then it will not be overwritten unless the `-f` option is given on the command line.

Robotic profiles

The RRT_Transactions_Patterns tables are scanned to set specific attributes on the transactions configured in the profiles. This table shows the mapping from situations to profiles. For every transaction configured in a profile that matches the Application Pattern and Transaction Pattern columns, the following values are set:

- Collect Instances
- Importance
- Minimum Response Time Threshold

In situation columns where a script name or playback command is defined, only the Equals (==) and In list (*IN) operators are recognized.

Table 74. Robotic situation attributes to profile mapping

Situation Attribute	Application Management Configuration Editor property
Situation Name	Profile Name [1]
Situation Description	Profile Description [2]
Sampling Interval	Script Interval [3]
Timeout Period	Timeout Period
Number Retries	Number of Retries
Minimum Response Time Threshold	Minimum Response Time Threshold
Retry Lag Time	Retry Lag Time
Importance	Importance
Collect Instances	Collect Instances
Sampling Percent	Always set to *100* for Robotic transactions.
Concurrent CLI Playback	Concurrent CLI Playback
Situation Distribution	Profile Distribution

Notes:

- [1] In the case where there are multiple situations to be condensed into a single profile, the profile name is taken from the situation name that comes first alphabetically (not lexicographically). You can manually change the profile name.
- [2] In the case where there are multiple situations to be condensed into a single profile, the first situation description is used.
- [3] The situation sampling interval is expressed in seconds, but the profile script interval accepts only minutes. Therefore, when converting this value from situations, it is rounded up or down to the nearest minute.

Non-robotic profiles

After the transactions have been created from all active

CRT_Transactions_Patterns and WRT_Transaction_Patterns situation tables, the profiles are created from information contained in the same tables. These situations are converted into as few profiles as possible, where all situations of each agent type with the same distribution list are grouped into a single profile.

Table 75. Non-robotic situation attributes to profile mapping

Situation Attribute	**Application Management Configuration Editor property**
Situation Name	Profile Name [4]
Situation Description	Profile Description [5]
Minimum Response Time Threshold	Minimum Response Time Threshold
Importance	Importance
Collect Instances	Collect Instances
Sampling Percent	Sampling Percent
Situation Distribution	Profile Distribution

Notes:

- [4] In the case where there are multiple situations to be condensed into a single profile, the profile name is taken from the situation name that comes first alphabetically (not lexicographically. You can manually change the profile name.
- [5] In the case where there are multiple situations to be condensed into a single profile, the first situation description is used.

Messages

When running the `sit2profile` utility, you can ignore a message that might be written to standard output, similar to the following example:

```
00000000,11/19/08 7:53 PM,009.048.132.108,00000000,VBJ-Application,main,ALERT,
Only limited VisiSecure functionality is enabled, If the product has a valid
license make sure all licensing related jar files are in the CLASSPATH
```

Appendix E. Tivoli Enterprise Console event mapping

Generic event mapping provides useful event class and attribute information for situations that do not have specific event mapping defined. Each event class corresponds to an attribute group in the monitoring agent. For a description of the event slots for each event class, see the tables described in this appendix. For more information about mapping attribute groups to event classes, see the Tivoli Enterprise Console product documentation.

Configuring the Tivoli Enterprise Console

To configure the Tivoli Enterprise Console, complete the following procedure:

1. Ensure sure that the TEC IF is configured to point to the correct Tivoli Enterprise Console server with the correct host port information.
2. Install the om_tec.baroc and kt<*n*>.baroc files into a rule base for Tivoli Enterprise Console and activate it.

   Note: When you install Tivoli Enterprise Monitoring Server support, the installation places the baroc files into one of the following directories, depending on your operating system:
   - On Windows systems: <*ITM_HOME*>\CMS\teclib
   - On UNIX systems: <*ITM_HOME*>/tables/<TEMS_NAME>/TECLIB

   The following kt<*n*>.baroc files are available:
   - kt3.baroc, for the Application Management Console event classes.
   - kt4.baroc, for the Client Response Time event classes.
   - kt5.baroc, for the Web Response Time event classes.
   - kt6.baroc, for the Robotic Response Time event classes.
3. In this same directory, edit the tecserver.txt file to add the situations for which you want to see events, using the following format:

   <*SituationName*>=*,SEVERITY=CRITICAL|WARNING|UNKNOWN

   For example, CRT_Response_Time_Threshold=*,SEVERITY=CRITICAL.
4. Restart the Tivoli Enterprise Monitoring Server.

Specifying attributes to include in events

Event data forwarded to Tivoli Enterprise Console is defined by the content of the following two file types in the Tivoli Enterprise Monitoring Server file system:

- Baroc files
- Event Mapping files

Baroc files, sometimes referred to as *event class definition* files, define different types of classes of events that an event server can receive. Event class definitions are generally structured as shown in the following example (keyword syntax is in upper case):

```
TEC_CLASS: class_name ISA super_class_name
DEFINES {
attribute_definitions;
};
END
```

The syntax for baroc files is case sensitive. The baroc files supplied with the various ITCAM for Transactions agents have *attribute_definitions* comprising an attribute name and data type. The following data types are used in these baroc files:

- STRING - A string value
- REAL - A real value
- ENUM - A value of an enumeration
- INT32 - A 32-bit integer value
- INTEGER - A 29-bit integer value

Detailed information regarding Event Classes and Attributes can be found in the Rule Developers Guide, located at the following website:`http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itecruledev.doc/ecodmst.htm`

Event Mapping files are XML files that specify how situation events data is mapped to baroc file event classes and attributes. Event mapping files are located in the same directory as the baroc files, and are similarly named:

- `kt3.map`, for the Application Management Console event classes.
- `kt4.map`, for the Client Response Time event classes.
- `kt5.map`, for the Web Response Time event classes.
- `kt6.map`, for the Robotic Response Time event classes.

The event mapping file has the following format:

```
<itmEventMapping:agent>
  <id>xx</id>
  <version>n.n</version>
  <event_mapping>
    <attributeTable>
      <class/>
      <slot>
        <mappedAttribute/>
        or
        <mappedAttributeEnum/>
      </slot>
      :.......... one or more slot tags
    </attributeTable>
    :.......... one or more attributeTable tags
  </event_mapping>
</itmeventMapping:agent>
```

The syntax and usage of the tags in the mapping file are described as follows:

<attributeTable>

Syntax: `<attributeTable name="`*`attribute_table_name`*`" [freeSpace="`*`nnnn`*`"]>`, where *nnnn* is an integer value.

Usage: The freeSpace="*nnnn*" parameter is the maximum free space available in the TEC event buffer for additional slots after all of the slots that are defined in this event map are built.

<slot> Syntax: `<slot name="`*`slot_name`*`">`

Usage: Defines a slot in the TEC event. The name of the slot is *slot_name*.

<mappedAttribute>

Syntax: `<mappedAttribute name="`*`attribute_name`*`" [multiplier="`*`nnn`*`"]>`

Usage: Specifies the attribute name that is mapped to the slot being defined. If *attribute_name* is not included in the event data, a null value is used. If the optional `multiplier=` parameter is specified and the value of the attribute is numeric, the value assigned for the slot is the attribute value multiplied by the number specified in the `multiplier=` parameter.

<mappedAttributeEnum>

Syntax: `<mappedAttributeEnum name="attribute_name">`

Usage: similar to the <mappedAttribute> tag, except if the attribute is defined as an enumerated value in the attribute file, the external enumerated string is used as the slot value instead of the attribute value. If there is no external enumerated string defined that matches the attribute value, the attribute value is used instead.

By default, not every attribute is included in events that are sent to Tivoli Enterprise Console. See the Attribute Groups section of the Response Time chapter in the *User's Guide* for an indication of which attributes in each group are preselected to be forwarded in events to Netcool/OMNIbus or Tivoli Enterprise Console.

Example: Forwarding additional Robotic Response Time attributes in events

You can modify the baroc and event mapping files to specify additional attributes to forward in events. The following example procedure illustrates how to modify the kt6.baroc and kt6.map files to create additional event slots for attributes that are not already being forwarded in events. This example will use attributes in the Robotic Response Time agent attribute group, RRT_Robotic_Playback_Events. You can use this same basic procedure to modify the baroc and event mapping files for any of the ITCAM for Transaction agents.

The procedure includes the following basic steps, which are further described below:

1. Determine the amount of free space available in the event mapping file.
2. Collect details about the name and size of the attributes to be added to the baroc and event mapping files.
3. Edit the baroc and event mapping files to create slots for the additional attributes.
4. Apply the baroc and event mapping file changes.

Step 1: Determine the amount of free space available in the event mapping file:

1. Open the following file in the Tivoli Enterprise Monitoring Server file system:
   - On Windows systems: `<ITM_HOME>\CMS\teclib\kt6.map`
   - On UNIX systems: `<ITM_HOME>/tables/<TEMS_NAME>/TECLIB/kt6.map`
2. Search for the string: `RRT_Robotic_Playback_Events`.
3. Make note of the *freeSpace* value, similar to the following example:

```
<attributeTable freeSpace="398" name="RRT_Robotic_Playback_Events">
  <class name="ITM_RRT_Robotic_Playback_Events"/>
  <slot slotName="application_name">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Application_Name"/>
  </slot>
  <slot slotName="transaction_name">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Transaction_Name"/>
  </slot>
  <slot slotName="kt6_situation_name">
```

```
    <mappedAttribute name="RRT_Robotic_Playback_Events.Situation_Name"/>
  </slot>
  <slot slotName="command_name">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Command_Name"/>
  </slot>
  <slot slotName="violation_data">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Violation_Data"/>
  </slot>
  <slot slotName="expected_data">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Expected_Data"/>
  </slot>
  <slot slotName="additional_details">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Additional_Details"/>
  </slot>
  <slot slotName="captured_content_location">
    <mappedAttribute
    name="RRT_Robotic_Playback_Events.Captured_Content_Location"/>
  </slot>
</attributeTable>
```

In this example, the attribute table has 398 bytes of available free space.

Step 2: Collect details about the name and size of the attributes to be added to the baroc and event mapping files:

1. In the Information Center or in the *ITCAM for Transactions User's Guide*, look up the RRT Robotic Playback Events attribute group to see a table similar to the following example:

Attribute (click on the link for a description)	Tivoli Data Warehouse term for historical reporting	Tivoli Data Warehouse database column size in bytes	Forwarded in events to Netcool/OMNIBus or Tivoli Enterprise Console
Additional Details	Additional_Details	1024	Yes
Application Name	Application_Name	128	Yes
Captured Content Location	Captured_Content _Location	512	Yes
Generic Playback Command	Command_Name	128	Yes
Event Timestamp	Event_Timestamp	16	No
Event Type	Event_Type	128	No
Expected Data	Expected_Data	128	Yes
Origin Node	Origin Node	32	No
Script Name	Script_Name	128	No
Script Type	Script_Type	128	No
Sample Timestamp	Sample_Timestamp	16	No
Situation Name	Situation_Name	128	Yes
Transaction Name	Transaction_Name	128	Yes
Violation Data	Violation_Data	128	Yes

In the far right column of the table, note that there are a number of attributes that are not, by default, forwarded in events. Suppose for this example that you want to begin forwarding data for the following attributes in events:

- Event_Type
- Script_Name
- Script_Type

2. In the table, note the *Tivoli Data Warehouse term* and the *column size in bytes* value for the attributes you want to add to the existing event. The total size of the attributes to be added must not be greater than the *freeSpace* value noted in the kt6.map file.

   For this example, the amount of free space is 398 bytes. From the above table, each of the attributes being added is 128 bytes, for a total of 384 bytes.

3. To add new entries to the kt6.map and kt6.baroc files, you need the *mappedAttribute* name and the *slotName*. The full *mappedAttribute* name is constructed from the Attribute Group Name (*RRT_Robotic_Playback_Events*) and the Attribute name from the **Tivoli Data Warehouse term for historical reporting** column of the above Attribute Group table. For our example these will be:

   - RRT_Robotic_Playback_Events.Event_Type
   - RRT_Robotic_Playback_Events.Script_Name
   - RRT_Robotic_Playback_Events.Script_Type

   The value for *slotName* is obtained from the Robotic Response Time Event classes table later in this appendix:

Table 76. Robotic Response Time event slots to event classes

Event class	Event slot

`ITM_RRT_Robotic_Playback_Events ISA KT6_Base`	`RRT_Robotic_Playback_Events` attribute group
	`origin_node: STRING;`
	`sample_timestamp: STRING;`
	`event_timestamp: STRING;`
	`script_name: STRING;`
	`script_type: STRING;`
	`script_type_enum: STRING;`
	`event_type: STRING;`
	`event_type_enum: STRING;`
	`kt6_situation_name: STRING;`
	`command_name: STRING;`
	`violation_data: STRING;`
	`expected_data: STRING;`
	`additional_details: STRING;`
	`application_name: STRING;`
	`transaction_name: STRING;`
	`captured_content_location: STRING;`

The values for *slotName* in baroc and map files are the same as the corresponding entry in the Attribute Group table, but all lower-case. For example, *Application_Name* from the Attribute Group table is *application_name* in the baroc file, and is the value for *slotName* in the map file. For our example, the following entries are present:

- event_type: STRING;
- script_name: STRING;
- script_type: STRING;

Step 3: Edit the baroc and event mapping files to create slots for the additional attributes

1. Make backup copies of the following files:
   - On Windows systems:
     - *<ITM_HOME>*\CMS\teclib\kt6.baroc
     - *<ITM_HOME>*\CMS\teclib\kt6.map
   - On UNIX and Linux systems:

- <*ITM_HOME*>/tables/tems_name/TECLIB/kt6.baroc
- <*ITM_HOME*>/tables/tems_name/TECLIB/kt6.map

2. Edit the `kt6.baroc` file to add the new attributes to the RRT_Robotic_Playback_Events Event Class.

   The resulting event class should look similar to the following example:

```
TEC_CLASS :
        ITM_RRT_Robotic_Playback_Events ISA KT6_Base
        DEFINES {

                #
                # RRT_Robotic_Playback_Events attribute group
                #

                kt6_situation_name: STRING;
                command_name: STRING;
                violation_data: STRING;
                expected_data: STRING;
                additional_details: STRING;
                application_name: STRING;
                transaction_name: STRING;
                captured_content_location: STRING;
                event_type: STRING;
                script_name: STRING;
                script_type: STRING;
        };
END
```

3. Edit the `kt6.map` file and for each of these three new attributes, add the following lines:

```
<slot slotName="xxxxxx">
  <mappedAttribute name="yyyyyy.zzzzzz"/>
</slot>
```

   xxxxxx The Event Slot name from the appropriate TEC Event Mapping table or the lower-case version of the Attribute name from the appropriate Attribute Group table.

   yyyyyy
   Attribute Group name

   zzzzzz Attribute name

   In addition, update the value of the *freeSpace* parameter to account for the new attributes added. In this example, the original value of *398* bytes is reduced by *384* bytes, to a resulting *freeSpace* value of *14* bytes.

   The resulting kt6.map entry should look similar to the following example:

```
<attributeTable freeSpace="14" name="RRT_Robotic_Playback_Events">
  <class name="ITM_RRT_Robotic_Playback_Events"/>
  <slot slotName="application_name">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Application_Name"/>
  </slot>
  <slot slotName="transaction_name">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Transaction_Name"/>
  </slot>
  <slot slotName="kt6_situation_name">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Situation_Name"/>
  </slot>
  <slot slotName="command_name">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Command_Name"/>
  </slot>
  <slot slotName="violation_data">
    <mappedAttribute name="RRT_Robotic_Playback_Events.Violation_Data"/>
  </slot>
  <slot slotName="expected_data">
```

```
  <mappedAttribute name="RRT_Robotic_Playback_Events.Expected_Data"/>
</slot>
<slot slotName="additional_details">
  <mappedAttribute name="RRT_Robotic_Playback_Events.Additional_Details"/>
</slot>
<slot slotName="captured_content_location">
  <mappedAttribute
  name="RRT_Robotic_Playback_Events.Captured_Content_Location"/>
</slot>
<slot slotName="event_type">
  <mappedAttribute name="RRT_Robotic_Playback_Events.Event_Type"/>
</slot>
<slot slotName="script_name">
  <mappedAttribute name="RRT_Robotic_Playback_Events.Script_Name"/>
</slot>
<slot slotName="script_type">
  <mappedAttribute name="RRT_Robotic_Playback_Events.Script_Type"/>
</slot></attributeTable>
```

Step 4: Apply the baroc and event mapping file changes

For the above changes to become effective you will have to perform the following steps:

1. Restart Tivoli Enterprise Monitoring Server. This ensures that the newly added attributes are included in events being forwarded.
2. To ensure the new attributes are recognized by the receiving event server, update the ITCAM for Transactions rule base on the TEC Server. For example:

```
wrb -delrbclass kt6.baroc -force ITCAMfT_RB
Copy the new baroc file from the TEMS file system to the TEC Server
wrb -imprbclass kt6.baroc ITCAMfT_RB
wrb -comprules ITCAMfT_RB
wrb -loadrb ITCAMfT_RB
wstopesvr
wstartesvr
```

Application Management Console Event classes

Each of the Application Management Console event classes is a child of the KT3_Base event class. The KT3_Base event class can be used for generic rules processing for any event from the Tivoli Enterprise Monitoring Agent.

Table 77. Application Management Console event slots to event classes

Event class	Event slot
ITM_DB_Agent_Details ISA KT3_Base	DB_Agent_Details attribute group timestamp: STRING; sample_time: STRING; origin_node: STRING; property: STRING; kt3_value: STRING; property_enum: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_ERT_Agent_Messages ISA KT3_Base	ERT_Agent_Messages attribute group origin_node: STRING; message_date_and_time: STRING; sample_timestamp: STRING; kt3_severity: INTEGER; kt3_severity_enum: STRING; component: STRING; component_enum: STRING; message_id: STRING; message_text: STRING; message_source: STRING;
ITM_DB_Application_Summary ISA KT3_Base	DB_Application_Summary attribute group origin_node: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_collector_type: STRING; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; application: STRING; appkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Client_Summary ISA KT3_Base	DB_Client_Summary attribute group origin_node: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_collector_type: STRING; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; client: STRING; ip: STRING; kt3_hostname: STRING; clientkey: STRING; maximum_response_time_threshold: REAL;
ITM_DB_Depot_Status ISA KT3_Base	DB_Depot_Status attribute group origin_node: STRING; last_update: STRING;
ITM_DB_File_Depot ISA KT3_Base	DB_File_Depot attribute group origin_node: STRING; attributes: STRING; path_arg: STRING; pattern_arg: STRING; kt3_server_path: STRING; file_type: STRING; desc: STRING; name: STRING; hidden: STRING; date_modified: STRING; file_size: STRING; checksum: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_T3_File_Transfer ISA KT3_Base	T3_File_Transfer attribute group origin_node: STRING; data: STRING; temp_id_pre: INTEGER; temp_id_post: INTEGER; kt3_server_path: STRING; xfer_mode: STRING; xfer_state_pre: STRING; xfer_state_post: STRING; fail_type: STRING; fail_code: INTEGER; file_mod1_pre: INTEGER; file_mod2_pre: INTEGER; file_mod1_post: INTEGER; file_mod2_post: INTEGER; offset_pre: INTEGER; offset_post: INTEGER; data_len_pre: INTEGER; data_len_post: INTEGER;
ITM_AMC_Internet_Service ISA KT3_Base	AMC_Internet_Service attribute group origin_node: STRING; overall_status: INTEGER; overall_status_enum: STRING; timestamp: STRING; good: INTEGER; marginal: INTEGER; failed: INTEGER; percent_good: REAL; percent_marginal: REAL; percent_failed: REAL; totaltime: REAL; percent_available: REAL; total_requests: INTEGER; profile: STRING; service: STRING; host: STRING; agent: STRING; profkey: STRING; description: STRING;
ITM_AMC_Internet_Service_Element ISA KT3_Base	AMC_Internet_Service_Element attribute group origin_node: STRING; overall_status: INTEGER; overall_status_enum: STRING; timestamp: STRING; good: INTEGER; marginal: INTEGER; failed: INTEGER; percent_good: REAL; percent_marginal: REAL; percent_failed: REAL; totaltime: REAL; percent_available: REAL; total_requests: INTEGER; profile: STRING; service: STRING; host: STRING; agent: STRING; profkey: STRING; description: STRING; identchecksum: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Internet_Service_Agent ISA KT3_Base	AMC_Internet_Service_Agent attribute group origin_node: STRING; overall_status: INTEGER; overall_status_enum: STRING; timestamp: STRING; good: INTEGER; marginal: INTEGER; failed: INTEGER; percent_good: REAL; percent_marginal: REAL; percent_failed: REAL; totaltime: REAL; percent_available: REAL; total_requests: INTEGER; profile: STRING; service: STRING; host: STRING; agent: STRING; profkey: STRING; description: STRING; identchecksum: STRING;
ITM_AMC_Internet_Services_Profiles ISA KT3_Base	AMC_Internet_Services_Profiles attribute group origin_node: STRING; overall_status: INTEGER; overall_status_enum: STRING; timestamp: STRING; good: INTEGER; marginal: INTEGER; failed: INTEGER; percent_good: REAL; percent_marginal: REAL; percent_failed: REAL; totaltime: REAL; percent_available: REAL; total_requests: INTEGER; profile: STRING; service: STRING; host: STRING; agent: STRING; profkey: STRING; aggby: INTEGER; aggby_enum: INTEGER;
ITM_AMC_ISM ISA KT3_Base	AMC_ISM attribute group origin_node: STRING; data_timespan: INTEGER; data_interval: INTEGER; last_updated: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Robotic_Playback_Status ISA KT3_Base	AMC_Robotic_Playback_Status attribute group origin_node: STRING; robotic_node: STRING; sample_timestamp: STRING; kt3_situation_name: STRING; script_name: STRING; script_type: STRING; script_type_enum: STRING; command_name: STRING; last_run_startime: STRING; last_updated: STRING; last_run_duration: REAL; last_run_status: STRING; last_run_status_enum: STRING; current_run_status: STRING; current_run_status_enum: STRING; application_name: STRING; transaction_name: STRING;
ITM_DB_Sub_Node_App_Client_Summary ISA KT3_Base	DB_Sub_Node_App_Client_Summary attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; application: STRING; client: STRING; ip: STRING; kt3_hostname: STRING; appkey: STRING; clientkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Agent ISA KT3_Base	AMC_Agent attribute group origin_node: STRING; data_collector_type: STRING; data_collector_type_enum: STRING; data_timespan: INTEGER; data_interval: INTEGER; overall_status: INTEGER; overall_status_enum: STRING; request_volume: INTEGER; request_volume_enum: STRING; current_requests: INTEGER; good_requests: INTEGER; slow_requests: INTEGER; bad_requests: INTEGER; min_requests: INTEGER; max_requests: INTEGER; average_requests: INTEGER; response_time: REAL; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; web_server: INTEGER; web_server_enum: STRING; app_server: INTEGER; app_server_enum: STRING; msg_status: INTEGER; msg_status_enum: STRING; back_status: INTEGER; back_status_enum: STRING; client_status: INTEGER; client_status_enum: STRING; agent: STRING; type: STRING; last_updated: STRING; timestamp: STRING; application: STRING; appkey: STRING; transaction: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Application ISA KT3_Base	AMC_Application attribute group origin_node: STRING; data_collector_type: STRING; data_collector_type_enum: STRING; data_timespan: INTEGER; data_interval: INTEGER; overall_status: INTEGER; overall_status_enum: STRING; request_volume: INTEGER; request_volume_enum: STRING; current_requests: INTEGER; good_requests: INTEGER; slow_requests: INTEGER; bad_requests: INTEGER; min_requests: INTEGER; max_requests: INTEGER; average_requests: INTEGER; response_time: REAL; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; web_server: INTEGER; web_server_enum: STRING; app_server: INTEGER; app_server_enum: STRING; msg_status: INTEGER; msg_status_enum: STRING; back_status: INTEGER; back_status_enum: STRING; client_status: INTEGER; client_status_enum: STRING; agent: STRING; type: STRING; last_updated: STRING; timestamp: STRING; application: STRING; appkey: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Sub_Node_Application_OverTime ISA KT3_Base	DB_Sub_Node_Application_OverTime attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; application: STRING; timestamp: STRING; appkey: STRING; maximum_response_time_threshold: REAL;
ITM_DB_Sub_Node_Application_Summary ISA KT3_Base	DB_Sub_Node_Application_Summary attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; application: STRING; appkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Sub_Node_App_Server_Summary ISA KT3_Base	DB_Sub_Node_App_Server_Summary attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; application: STRING; server: STRING; ip: STRING; appkey: STRING; serverkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Client ISA KT3_Base	AMC_Client attribute group origin_node: STRING; data_collector_type: STRING; data_collector_type_enum: STRING; data_timespan: INTEGER; data_interval: INTEGER; overall_status: INTEGER; overall_status_enum: STRING; request_volume: INTEGER; request_volume_enum: STRING; current_requests: INTEGER; good_requests: INTEGER; slow_requests: INTEGER; bad_requests: INTEGER; min_requests: INTEGER; max_requests: INTEGER; average_requests: INTEGER; response_time: REAL; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; web_server: INTEGER; web_server_enum: STRING; app_server: INTEGER; app_server_enum: STRING; msg_status: INTEGER; msg_status_enum: STRING; back_status: INTEGER; back_status_enum: STRING; client_status: INTEGER; client_status_enum: STRING; agent: STRING; type: STRING; last_updated: STRING; timestamp: STRING; application: STRING; appkey: STRING; client: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Sub_Node_Client_Over_Time ISA KT3_Base	DB_Sub_Node_Client_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; client: STRING; ip: STRING; kt3_hostname: STRING; timestamp: STRING; clientkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Sub_Node_Client_Summary ISA KT3_Base	DB_Sub_Node_Client_Summary attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; client: STRING; ip: STRING; kt3_hostname: STRING; clientkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Client_Agents ISA KT3_Base	AMC_Client_Agents attribute group origin_node: STRING; data_collector_type: STRING; data_collector_type_enum: STRING; data_timespan: INTEGER; data_interval: INTEGER; overall_status: INTEGER; overall_status_enum: STRING; request_volume: INTEGER; request_volume_enum: STRING; current_requests: INTEGER; good_requests: INTEGER; slow_requests: INTEGER; bad_requests: INTEGER; min_requests: INTEGER; max_requests: INTEGER; average_requests: INTEGER; response_time: REAL; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; web_server: INTEGER; web_server_enum: STRING; app_server: INTEGER; app_server_enum: STRING; msg_status: INTEGER; msg_status_enum: STRING; back_status: INTEGER; back_status_enum: STRING; client_status: INTEGER; client_status_enum: STRING; agent: STRING; type: STRING; last_updated: STRING; timestamp: STRING; application: STRING; appkey: STRING; client: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Sub_Node_Client_Server_Summary ISA KT3_Base	DB_Sub_Node_Client_Server_Summary attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; client: STRING; ip: STRING; kt3_hostname: STRING; server: STRING; server_ip: STRING; clientkey: STRING; serverkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Server ISA KT3_Base	AMC_Server attribute group origin_node: STRING; data_collector_type: STRING; data_collector_type_enum: STRING; data_timespan: INTEGER; data_interval: INTEGER; overall_status: INTEGER; overall_status_enum: STRING; request_volume: INTEGER; request_volume_enum: STRING; current_requests: INTEGER; good_requests: INTEGER; slow_requests: INTEGER; bad_requests: INTEGER; min_requests: INTEGER; max_requests: INTEGER; average_requests: INTEGER; response_time: REAL; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; web_server: INTEGER; web_server_enum: STRING; app_server: INTEGER; app_server_enum: STRING; msg_status: INTEGER; msg_status_enum: STRING; back_status: INTEGER; back_status_enum: STRING; server_status: INTEGER; server_status_enum: STRING; agent: STRING; type: STRING; last_updated: STRING; timestamp: STRING; application: STRING; appkey: STRING; server: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Sub_Node_Server_Over_Time ISA KT3_Base	DB_Sub_Node_Server_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; server: STRING; ip: STRING; timestamp: STRING; serverkey: STRING; maximum_response_time_threshold: REAL;
ITM_DB_Sub_Node_Server_Summary ISA KT3_Base	DB_Sub_Node_Server_Summary attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; server: STRING; ip: STRING; serverkey: STRING; maximum_response_time_threshold: REAL;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Server_Agents ISA KT3_Base	AMC_Server_Agents attribute group origin_node: STRING; data_collector_type: STRING; data_collector_type_enum: STRING; data_timespan: INTEGER; data_interval: INTEGER; overall_status: INTEGER; overall_status_enum: STRING; request_volume: INTEGER; request_volume_enum: STRING; current_requests: INTEGER; good_requests: INTEGER; slow_requests: INTEGER; bad_requests: INTEGER; min_requests: INTEGER; max_requests: INTEGER; average_requests: INTEGER; response_time: REAL; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; web_server: INTEGER; web_server_enum: STRING; app_server: INTEGER; app_server_enum: STRING; msg_status: INTEGER; msg_status_enum: STRING; back_status: INTEGER; back_status_enum: STRING; server_status: INTEGER; server_status_enum: STRING; agent: STRING; type: STRING; last_updated: STRING; timestamp: STRING; application: STRING; appkey: STRING; server: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_AMC_Transaction ISA KT3_Base	AMC_Transaction attribute group origin_node: STRING; data_collector_type: STRING; data_collector_type_enum: STRING; data_timespan: INTEGER; data_interval: INTEGER; overall_status: INTEGER; overall_status_enum: STRING; request_volume: INTEGER; request_volume_enum: STRING; current_requests: INTEGER; good_requests: INTEGER; slow_requests: INTEGER; bad_requests: INTEGER; min_requests: INTEGER; max_requests: INTEGER; average_requests: INTEGER; response_time: REAL; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; web_server: INTEGER; web_server_enum: STRING; app_server: INTEGER; app_server_enum: STRING; msg_status: INTEGER; msg_status_enum: STRING; back_status: INTEGER; back_status_enum: STRING; client_status: INTEGER; client_status_enum: STRING; agent: STRING; type: STRING; last_updated: STRING; timestamp: STRING; application: STRING; appkey: STRING; transaction: STRING;

Table 77. Application Management Console event slots to event classes (continued)

Event class	Event slot
ITM_DB_Sub_Node_Transaction_Summary ISA KT3_Base	DB_Sub_Node_Transaction_Summary attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; application: STRING; transaction: STRING; appkey: STRING; maximum_response_time_threshold: REAL;
ITM_DB_Sub_Node_Transaction_OverTime ISA KT3_Base	DB_Sub_Node_Transaction_OverTime attribute group origin_node: STRING; data_collector_type: STRING; aggby: INTEGER; aggby_enum: INTEGER; data_timespan: INTEGER; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; overall_time: REAL; successful_requests: INTEGER; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; type: STRING; data_timezoneoffset: INTEGER; scope: INTEGER; scope_enum: STRING; application: STRING; timestamp: STRING; transaction: STRING; appkey: STRING; maximum_response_time_threshold: REAL;

Client Response Time Event classes

Each of the event classes is a child of KT4_Base. The KT4_Base event class can be used for generic rules processing for any event from the Tivoli Enterprise Monitoring Agent.

Table 78. Client Response Time event classes to event slots

Event class	Event slot
ITM_CRT_Agent_Details ISA KT4_Base	CRT_Agent_Details attribute group timestamp: STRING; sample_time: STRING; origin_node: STRING; property: STRING; kt4_value: STRING; property_enum: STRING;
ITM_CRT_Agent_Messages ISA KT4_Base	CRT_Agent_Messages attribute group origin_node: STRING; sample_timestamp: STRING; message_date_and_time: STRING; kt4_severity: INTEGER; kt4_severity_enum: STRING; component: STRING; component_enum: STRING; message_id: STRING; message_text: STRING; message_source: STRING;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Application_Status ISA KT4_Base	CRT_Application_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; kt4_situation_status: INTEGER; kt4_situation_status_enum: INTEGER; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; scope: INTEGER; scope_enum: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Application_Over_Time ISA KT4_Base	CRT_Application_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Application_Summary ISA KT4_Base	CRT_Application_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Client_Application ISA KT4_Base	CRT_Client_Application attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; application: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Client_Over_Time ISA KT4_Base	CRT_Client_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; ip: STRING; kt4_hostname: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; ipv6: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;
ITM_CRT_Client_Patterns ISA KT4_Base	CRT_Client_Patterns attribute group origin_node: STRING; sample_time: STRING; timestamp: STRING; client_name: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; aggregates_uniquely: INTEGER; aggregates_uniquely_enum: STRING;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Client_Summary ISA KT4_Base	CRT_Client_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; ip: STRING; kt4_hostname: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; ipv6: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Client_Server ISA KT4_Base	CRT_Client_Server attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; server: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;
ITM_CRT_Depot_Status ISA KT4_Base	CRT_Depot_Status attribute group origin_node: STRING; depot_node: STRING;
ITM_CRT_Profile_Configuration ISA KT4_Base	CRT_Profile_Configuration attribute group origin_node: STRING; sample_timestamp: STRING; config_type: INTEGER; config_type_enum: STRING; config_name: STRING; entry_type: INTEGER; entry_type_enum: STRING; key_name: STRING; kt4_value: STRING;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Server_Application ISA KT4_Base	CRT_Server_Application attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; server: STRING; application: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Server_Status ISA KT4_Base	CRT_Server_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; kt4_situation_status: INTEGER; kt4_situation_status_enum: INTEGER; server: STRING; ip: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; ipv6: STRING; scope: INTEGER; scope_enum: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Server_Over_Time ISA KT4_Base	CRT_Server_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; server: STRING; ip: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; ipv6: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Server_Summary ISA KT4_Base	CRT_Server_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; server: STRING; ip: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; ipv6: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_SubTransaction_Status ISA KT4_Base	CRT_SubTransaction_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; kt4_situation_status: INTEGER; kt4_situation_status_enum: STRING; scope: INTEGER; scope_enum: STRING; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_SubTransaction_Instance ISA KT4_Base	CRT_SubTransaction_Instance attribute group origin_node: STRING; application_pattern: STRING; transaction_pattern: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; data_collector_type: STRING; sample_time: STRING; timestamp: STRING; application_name: STRING; transaction_name: STRING; client_name: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; instanceroot: STRING; response_time: REAL; status_code: INTEGER; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;
ITM_CRT_SubTransaction_Over_Time ISA KT4_Base	CRT_SubTransaction_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_SubTransaction_Summary ISA KT4_Base	CRT_SubTransaction_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Transaction_Status ISA KT4_Base	CRT_Transaction_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; kt4_situation_status: INTEGER; kt4_situation_status_enum: INTEGER; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; scope: INTEGER; scope_enum: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Transaction_Instance ISA KT4_Base	CRT_Transaction_Instance attribute group origin_node: STRING; application_pattern: STRING; transaction_pattern: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; data_collector_type: STRING; sample_time: STRING; aggby: INTEGER; aggby_enum: INTEGER; timestamp: STRING; application_name: STRING; transaction_name: STRING; client_name: STRING; rootuuid: STRING; instanceroot: STRING; response_time: REAL; status_code: INTEGER; minimum_response_time_threshold: REAL; importance: INTEGER; importance_enum: STRING; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; server: STRING; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Transaction_Over_Time ISA KT4_Base	CRT_Transaction_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;
ITM_CRT_Transactions_Patterns ISA KT4_Base	CRT_Transactions_Patterns attribute group origin_node: STRING; sample_time: STRING; timestamp: STRING; application_name: STRING; application_pattern: STRING; transaction_name: STRING; transaction_pattern: STRING; aggregate_applications_uniquely: INTEGER; aggregate_applications_uniquely_enum: STRING; aggregate_transactions_uniquely: INTEGER; aggregate_transactions_uniquely_enum: STRING; collect_instances: INTEGER; collect_instances_enum: STRING; sampling_percent: INTEGER; importance: INTEGER; importance_enum: STRING; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL;

Table 78. Client Response Time event classes to event slots (continued)

Event class	Event slot
ITM_CRT_Transaction_Summary ISA KT4_Base	CRT_Transaction_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; total_connect_time: REAL; total_download_time: REAL; number_browser_connections: INTEGER; average_connect_time: REAL; average_download_time: REAL; maximum_response_time_threshold: REAL; bytes_received: INTEGER; bytes_sent: INTEGER;

Web Response Time Event classes

Each of the event classes is a child of KT5_Base. The KT5_Base event class can be used for generic rules processing for any event from the Tivoli Enterprise Monitoring Agent.

Table 79. Web Response Time event classes to event slots

Event class	Event slot
ITM_WRT_Agent_Details ISA KT5_Base	WRT_Agent_Details attribute group origin_node: STRING; timestamp: STRING; property: STRING; property_enum: STRING; kt5_value: STRING; sample_time: STRING;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Agent_Messages ISA KT5_Base	WRT_Agent_Messages attribute group origin_node: STRING; message_date_and_time: STRING; sample_timestamp: STRING; kt5_severity: INTEGER; kt5_severity_enum: STRING; component: STRING; component_enum: STRING; message_id: STRING; message_text: STRING; message_source: STRING;
ITM_WRT_Application_Status ISA KT5_Base	WRT_Application_Status attribute group origin_node: STRING; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; sample_time: STRING; rank: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued in next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Application_Status ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; kt5_situation_status: INTEGER; kt5_situation_status_enum: INTEGER; scope: INTEGER; scope_enum: STRING; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; number_active_sessions: INTEGER; number_good_sessions: INTEGER; number_slow_sessions: INTEGER; number_failed_sessions: INTEGER; average_session_duration: INTEGER; average_page_views_per_session: INTEGER; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Application_Over_Time ISA KT5_Base	WRT_Application_Over_Time attribute group origin_node: STRING; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued in next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Application_Over_Time ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; total_users: INTEGER; total_good_users: INTEGER; total_slow_users: INTEGER; total_failed_users: INTEGER; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Application_Summary ISA KT5_Base	WRT_Application_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; (continued in next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Application_Summary ISA KT5_Base (continued)	(continued from previous row) request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; total_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Application ISA KT5_Base	WRT_Client_Application attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; application: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; (continued in next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Application ISA KT5_Base (Continued)	(continued from previous row) reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Status ISA KT5_Base	WRT_Client_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; client: STRING; ip: STRING; kt5_hostname: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; (continued in next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Status ISA KT5_Base (Continued)	(continued from previous row) request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; ipv6: STRING; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; kt5_situation_status: INTEGER; kt5_situation_status_enum: STRING; scope: INTEGER; scope_enum: STRING; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Over_Time ISA KT5_Base	WRT_Client_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; ip: STRING; kt5_hostname: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Over_Time ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; ipv6: STRING; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; total_users: INTEGER; total_good_users: INTEGER; total_slow_users: INTEGER; total_failed_users: INTEGER; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;
ITM_WRT_Client_Patterns ISA KT5_Base	WRT_Client_Patterns attribute group origin_node: STRING; timestamp: STRING; client_name: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; aggregates_uniquely: INTEGER; aggregates_uniquely_enum: STRING; sample_time: STRING;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Summary ISA KT5_Base	WRT_Client_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; ip: STRING; kt5_hostname: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Summary ISA KT5_Base (Continued)	(continued from previous row) average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; ipv6: STRING; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Client_Server ISA KT5_Base	WRT_Client_Server attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; server: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Depot_Status ISA KT5_Base	WRT_Depot_Status attribute group origin_node: STRING; depot_node: STRING;
ITM_WRT_Profile_Configuration ISA KT5_Base	WRT_Profile_Configuration attribute group origin_node: STRING; sample_timestamp: STRING; config_type: INTEGER; config_type_enum: STRING; config_name: STRING; entry_type: INTEGER; entry_type_enum: STRING; key_name: STRING; kt5_value: STRING;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Server_Application ISA KT5_Base	WRT_Server_Application attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; server: STRING; application: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Server_Status ISA KT5_Base	WRT_Server_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; kt5_situation_status: INTEGER; kt5_situation_status_enum: INTEGER; server: STRING; ip: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Server_Status ISA KT5_Base (continued)	(continued from previous row) redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; ipv6: STRING; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; scope: INTEGER; scope_enum: STRING; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_active_sessions: INTEGER; number_good_sessions: INTEGER; number_slow_sessions: INTEGER; number_failed_sessions: INTEGER; average_session_duration: INTEGER; average_page_views_per_session: INTEGER; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER; number_of_ssl_errors: INTEGER; number_of_ssl_warnings: INTEGER; number_of_network_ssl_errors: INTEGER; number_of_server_ssl_errors: INTEGER; number_of_server_ssl_warnings: INTEGER; number_of_client_ssl_errors: INTEGER; number_of_client_ssl_warnings: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Server_Over_Time ISA KT5_Base	WRT_Server_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; server: STRING; ip: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Server_Over_Time ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; ipv6: STRING; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; total_users: INTEGER; total_good_users: INTEGER; total_slow_users: INTEGER; total_failed_users: INTEGER; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER; number_of_ssl_errors: INTEGER; number_of_ssl_warnings: INTEGER; number_of_network_ssl_errors: INTEGER; number_of_server_ssl_errors: INTEGER; number_of_server_ssl_warnings: INTEGER; number_of_client_ssl_errors: INTEGER; number_of_client_ssl_warnings: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Server_Summary ISA KT5_Base	WRT_Server_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; server: STRING; ip: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Server_Summary ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; ipv6: STRING; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER; number_of_ssl_errors: INTEGER; number_of_ssl_warnings: INTEGER; number_of_network_ssl_errors: INTEGER; number_of_server_ssl_errors: INTEGER; number_of_server_ssl_warnings: INTEGER; number_of_client_ssl_errors: INTEGER; number_of_client_ssl_warnings: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_SubTransaction_Status ISA KT5_Base	WRT_SubTransaction_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; kt5_situation_status: INTEGER; kt5_situation_status_enum: INTEGER; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_SubTransaction_Status ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; scope: INTEGER; scope_enum: STRING; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_SubTransaction_Instance ISA KT5_Base	WRT_SubTransaction_Instance attribute group origin_node: STRING; application_pattern: STRING; transaction_pattern: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; data_collector_type: STRING; sample_time: STRING; gmt_offset: INTEGER; timezone: STRING; referrer_url: STRING; browser_description: STRING; server_description: STRING; url_hostname: STRING; url_path: STRING; url_file: STRING; url_query_string: STRING; url_anchor: STRING; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; timestamp: STRING; application_name: STRING; transaction_name: STRING; client_name: STRING; rootuuid: STRING; response_time: REAL; status_code: INTEGER; ip_source_address: STRING; ip_destination_address: STRING; ip_destination_port: INTEGER; url: STRING; application_protocol: STRING; page_title: STRING; method: INTEGER; method_enum: STRING; parentuuid: STRING; currentuuid: STRING; instanceroot: STRING; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_SubTransaction_Over_Time ISA KT5_Base	WRT_SubTransaction_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; total_users: INTEGER; total_good_users: INTEGER; total_slow_users: INTEGER; total_failed_users: INTEGER; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_SubTransaction_Summary ISA KT5_Base	WRT_SubTransaction_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Status ISA KT5_Base	WRT_Transaction_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; kt5_situation_status: INTEGER; kt5_situation_status_enum: INTEGER; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Status ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; scope: INTEGER; scope_enum: STRING; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Instance ISA KT5_Base	WRT_Transaction_Instance attribute group origin_node: STRING; application_pattern: STRING; transaction_pattern: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; minimum_response_time_threshold: REAL; data_collector_type: STRING; importance: INTEGER; importance_enum: STRING; sample_time: STRING; gmt_offset: INTEGER; timezone: STRING; referrer_url: STRING; browser_description: STRING; server_description: STRING; application_protocol: STRING; url_hostname: STRING; url_path: STRING; url_file: STRING; url_query_string: STRING; url_anchor: STRING; method: INTEGER; method_enum: STRING; aggby: INTEGER; aggby_enum: INTEGER; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; request_bytes: INTEGER; reply_bytes: INTEGER; total_object_count: INTEGER; total_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; render_time: REAL; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Instance ISA KT5_Base (Continued)	(continued from previous row) timestamp: STRING; application_name: STRING; transaction_name: STRING; client_name: STRING; rootuuid: STRING; response_time: REAL; status_code: INTEGER; ip_source_address: STRING; ip_destination_address: STRING; ip_destination_port: INTEGER; url: STRING; page_title: STRING; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; total_bytes: INTEGER; instanceroot: STRING; server: STRING; maximum_response_time_threshold: REAL; user: STRING; session: STRING; sort_order: INTEGER; sort_order_enum: STRING; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; content_error_type: INTEGER; content_error_type_enum: STRING; content_error_search_string: STRING;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Over_Time ISA KT5_Base	WRT_Transaction_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Over_Time ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; total_users: INTEGER; total_good_users: INTEGER; total_slow_users: INTEGER; total_failed_users: INTEGER; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;
ITM_WRT_Transactions_Patterns ISA KT5_Base	WRT_Transactions_Patterns attribute group origin_node: STRING; sample_time: STRING; timestamp: STRING; application_name: STRING; application_pattern: STRING; transaction_name: STRING; transaction_pattern: STRING; aggregate_applications_uniquely: INTEGER; aggregate_applications_uniquely_enum: STRING; aggregate_transactions_uniquely: INTEGER; aggregate_transactions_uniquely_enum: STRING; collect_instances: INTEGER; collect_instances_enum: STRING; sampling_percent: INTEGER; importance: INTEGER; importance_enum: STRING; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL;

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Summary ISA KT5_Base	WRT_Transaction_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_Transaction_Summary ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; average_users: REAL; average_good_users: REAL; average_slow_users: REAL; average_failed_users: REAL; maximum_response_time_threshold: REAL; number_of_retransmissions: INTEGER; kilobytes_retransmitted: INTEGER; number_of_content_check_errors: INTEGER;
ITM_WRT_User_Sessions ISA KT5_Base	WRT_User_Sessions attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; application: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; average_client_time: REAL; average_load_time: REAL; average_network_time: REAL; average_server_time: REAL; average_resolve_time: REAL; request_packet_count: INTEGER; request_ack_packet_count: INTEGER; reply_packet_count: INTEGER; reply_ack_packet_count: INTEGER; (continued on next row)

Table 79. Web Response Time event classes to event slots (continued)

Event class	Event slot
ITM_WRT_User_Sessions ISA KT5_Base (Continued)	(continued from previous row) total_kbytes: INTEGER; request_kbytes: INTEGER; reply_kbytes: INTEGER; average_object_count: INTEGER; average_object_size: INTEGER; total_header_requests_count: INTEGER; total_header_request_resolve_time: REAL; informational: INTEGER; successes: INTEGER; redirections: INTEGER; client_errors: INTEGER; server_errors: INTEGER; number_of_403s: INTEGER; number_of_404s: INTEGER; number_of_500s: INTEGER; percent_informational: REAL; percent_successes: REAL; percent_redirections: REAL; percent_client_errors: REAL; percent_server_errors: REAL; percent_of_403s: REAL; percent_of_404s: REAL; percent_of_500s: REAL; average_render_time: REAL; maximum_response_time_threshold: REAL; number_of_sessions: INTEGER; number_of_requests: INTEGER; scope: INTEGER; scope_enum: STRING; user: STRING; session: STRING; session_start_time: STRING; session_end_time: STRING; session_duration: REAL; number_of_content_check_errors: INTEGER;
ITM_WRT_SSL_Alert_Current_Status ISA KT5_Base	WRT_SSL_Alert_Current_Status attribute group alert_name: STRING; alert_type: INTEGER; alert_type_enum: STRING; client_group: STRING; count: INTEGER; first_occurrence: STRING; origin_node: STRING; sample_time: STRING; scope: INTEGER; server_ip: STRING; server_port: INTEGER; severity: INTEGER; severity_enum: STRING; timestamp: STRING;

Robotic Response Time Event classes

Each of the event classes is a child of KT6_Base. The KT6_Base event class can be used for generic rules processing for any event from the Tivoli Enterprise Monitoring Agent.

Table 80. Robotic Response Time event slots to event classes

Event class	Event slot
ITM_RRT_Agent_Details ISA KT6_Base	RRT_Agent_Details attribute group timestamp: STRING; sample_time: STRING; origin_node: STRING; property: STRING; kt6_value: STRING; property_enum: STRING;
ITM_RRT_Agent_Messages ISA KT6_Base	RRT_Agent_Messages attribute group origin_node: STRING; sample_timestamp: STRING; message_date_and_time: STRING; kt6_severity: INTEGER; kt6_severity_enum: STRING; component: STRING; component_enum: STRING; message_id: STRING; message_text: STRING; message_source: STRING;
ITM_RRT_Application_Status ISA KT6_Base	RRT_Application_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; kt6_situation_status: INTEGER; kt6_situation_status_enum: INTEGER; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; scope: INTEGER; scope_enum: STRING; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Application_Over_Time ISA KT6_Base	RRT_Application_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Application_Summary ISA KT6_Base	RRT_Application_Summary attribute group origin_node: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Client_Application ISA KT6_Base	RRT_Client_Application attribute group origin_node: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; application: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Client_Over_Time ISA KT6_Base	RRT_Client_Over_Time attribute group origin_node: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; ip: STRING; kt6_hostname: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; ipv6: STRING; maximum_response_time_threshold: REAL;
ITM_RRT_Client_Patterns ISA KT6_Base	RRT_Client_Patterns attribute group origin_node: STRING; timestamp: STRING; client_name: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; aggregates_uniquely: INTEGER; aggregates_uniquely_enum: STRING; sample_time: STRING;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Client_Summary ISA KT6_Base	RRT_Client_Summary attribute group origin_node: STRING; sample_time: STRING; rank: REAL; aggby: INTEGER; aggby_enum: INTEGER; client: STRING; ip: STRING; kt6_hostname: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; data_collector_type: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; ipv6: STRING; maximum_response_time_threshold: REAL;
ITM_RRT_Depot_Status ISA KT6_Base	RRT_Depot_Status attribute group origin_node: STRING; depot_node: STRING;
ITM_RRT_Robotic_Playback_Events_Sampled ISA KT6_Base	RRT_Robotic_Playback_Events_Sampled attribute group origin_node: STRING; sample_timestamp: STRING; event_timestamp: STRING; script_type: STRING; script_type_enum: STRING; event_type: STRING; event_type_enum: STRING; kt6_situation_name: STRING; script_name: STRING; command_name: STRING; violation_data: STRING; expected_data: STRING; additional_details: STRING; application_name: STRING; transaction_name: STRING; captured_content_location: STRING;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Robotic_Playback_Configuration ISA KT6_Base	RRT_Robotic_Playback_Configuration attribute group origin_node: STRING; sample_timestamp: STRING; robotic_script_name: STRING; script_type: STRING; script_type_enum: STRING; generic_playback_command: STRING; timeout_period: INTEGER; number_retries: INTEGER; retry_lag_time: INTEGER; concurrent: STRING; concurrent_enum: STRING; abort_on_violation: STRING; abort_on_violation_enum: STRING; cli_success_rc: INTEGER;
ITM_RRT_Robotic_Playback_Events ISA KT6_Base	RRT_Robotic_Playback_Events attribute group origin_node: STRING; sample_timestamp: STRING; event_timestamp: STRING; script_name: STRING; script_type: STRING; script_type_enum: STRING; event_type: STRING; event_type_enum: STRING; kt6_situation_name: STRING; command_name: STRING; violation_data: STRING; expected_data: STRING; additional_details: STRING; application_name: STRING; transaction_name: STRING; captured_content_location: STRING;
ITM_RRT_Robotic_Playback_Status ISA KT6_Base	RRT_Robotic_Playback_Status attribute group origin_node: STRING; sample_timestamp: STRING; kt6_situation_name: STRING; script_name: STRING; script_type: STRING; script_type_enum: STRING; command_name: STRING; last_run_startime: STRING; last_updated: STRING; last_run_duration: REAL; last_run_status: STRING; last_run_status_enum: STRING; current_run_status: STRING; current_run_status_enum: STRING; application_name: STRING; transaction_name: STRING;
ITM_RRT_Profile_Configuration ISA KT6_Base	RRT_Profile_Configuration attribute group origin_node: STRING; sample_timestamp: STRING; config_type: INTEGER; config_type_enum: STRING; config_name: STRING; entry_type: INTEGER; entry_type_enum: STRING; key_name: STRING; kt6_value: STRING;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Realms ISA KT6_Base	RRT_Realms attribute group origin_node: STRING; timestamp: STRING; realm_name: STRING; host_name: STRING; user_name: STRING; password: STRING; realm_type: STRING; realm_type_enum: STRING; auth_type: STRING;
ITM_RRT_SubTransaction_Status ISA KT6_Base	RRT_SubTransaction_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; kt6_situation_status: INTEGER; kt6_situation_status_enum: INTEGER; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; scope: INTEGER; scope_enum: STRING; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_SubTransaction_Instance ISA KT6_Base	RRT_SubTransaction_Instance attribute group origin_node: STRING; application_pattern: STRING; transaction_pattern: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; data_collector_type: STRING; sample_time: STRING; timestamp: STRING; application_name: STRING; transaction_name: STRING; client_name: STRING; rootuuid: STRING; response_time: REAL; status_code: INTEGER; parentuuid: STRING; currentuuid: STRING; instanceroot: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_SubTransaction_Over_Time ISA KT6_Base	RRT_SubTransaction_Over_Time attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_SubTransaction_Summary ISA KT6_Base	RRT_SubTransaction_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; rank: REAL; transaction: STRING; rootuuid: STRING; parentuuid: STRING; currentuuid: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time: REAL; maximum_response_time: REAL; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Transaction_Status ISA KT6_Base	RRT_Transaction_Status attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; kt6_situation_status: INTEGER; kt6_situation_status_enum: INTEGER; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; rank: REAL; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; scope: INTEGER; scope_enum: STRING; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Transaction_Instance ISA KT6_Base	RRT_Transaction_Instance attribute group origin_node: STRING; application_pattern: STRING; transaction_pattern: STRING; client_hostname_pattern: STRING; client_ip_pattern: STRING; data_collector_type: STRING; sample_time: STRING; aggby: INTEGER; aggby_enum: INTEGER; timestamp: STRING; application_name: STRING; transaction_name: STRING; client_name: STRING; rootuuid: STRING; instanceroot: STRING; response_time: REAL; status_code: INTEGER; minimum_response_time_threshold: REAL; importance: INTEGER; importance_enum: STRING; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; server: STRING; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Transaction_Over_Time ISA KT6_Base	RRT_Transaction_Over_Time attribute group origin_node: STRING; sample_time: STRING; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; rank: REAL; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; maximum_response_time_threshold: REAL;
ITM_RRT_Transactions_Patterns ISA KT6_Base	RRT_Transactions_Patterns attribute group origin_node: STRING; sample_time: STRING; timestamp: STRING; application_name: STRING; application_pattern: STRING; transaction_name: STRING; transaction_pattern: STRING; aggregate_applications_uniquely: INTEGER; aggregate_applications_uniquely_enum: STRING; aggregate_transactions_uniquely: INTEGER; aggregate_transactions_uniquely_enum: STRING; collect_instances: INTEGER; collect_instances_enum: STRING; sampling_percent: INTEGER; importance: INTEGER; importance_enum: STRING; minimum_response_time_threshold: REAL; maximum_response_time_threshold: REAL;

Table 80. Robotic Response Time event slots to event classes (continued)

Event class	Event slot
ITM_RRT_Transaction_Summary ISA KT6_Base	RRT_Transaction_Summary attribute group origin_node: STRING; data_collector_type: STRING; sample_time: STRING; aggby: INTEGER; aggby_enum: INTEGER; application: STRING; transaction: STRING; client: STRING; server: STRING; rootuuid: STRING; importance: INTEGER; importance_enum: STRING; timestamp: STRING; data_interval: INTEGER; percent_failed: REAL; percent_slow: REAL; percent_good: REAL; percent_available: REAL; average_response_time: REAL; failed_requests: INTEGER; total_requests: INTEGER; slow_requests: INTEGER; good_requests: INTEGER; minimum_response_time_threshold: REAL; minimum_response_time: REAL; maximum_response_time: REAL; rank: REAL; total_server_response_time: REAL; total_connect_time: REAL; total_dns_time: REAL; total_resolve_time: REAL; average_server_response_time: REAL; average_connect_time: REAL; average_dns_time: REAL; average_resolve_time: REAL; client_time: REAL; network_time: REAL; server_time: REAL; maximum_response_time_threshold: REAL;

Appendix F. Instrumenting ARM

The Application Response Measurement (ARM) API is a set of standard API calls that enable you to measure the performance of your applications.

The ARM API is mainly used to measure response time, but can also be used to record application availability. For more information on ARM, see the following documentation resources:

- The Open Group ARM documentation is a good primary source of information on the API:

  http://www.opengroup.org/arm/

  The ARM 4.0 standard documentation (ARM 4.0 Version 2 C Language Binding Technical Standard) is the definitive guide for the ARM 4.0 API and is very complete.
- For the ARM 4.0 API, see the EWLM-based documentation:

  http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicaw/eicaaarmregmetric.htm
- *Application Response Measurement Instrumentation Guide* (September 2007):

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.itcamwas_rt.doc_6.6/ARM_Instr_Guide.pdf

  This ARM Instrumentation Guide is based on ARM v2.0. Although this version of ARM has been superseded by ARM v4.0, the information in this guide is still relevant. The guide describes how to instrument ARM for applications so that they can interact with ITCAM products.
- IBM technote describing ARM Error codes for ITCAM for RTT 6.1 and ITCAM for RT 6.2:

  http://www-01.ibm.com/support/docview.wss?uid=swg21268175

How the ARM API works

To measure response time for an application, the application must be instrumented. Instrumenting an application means including ARM API calls in the application source code.

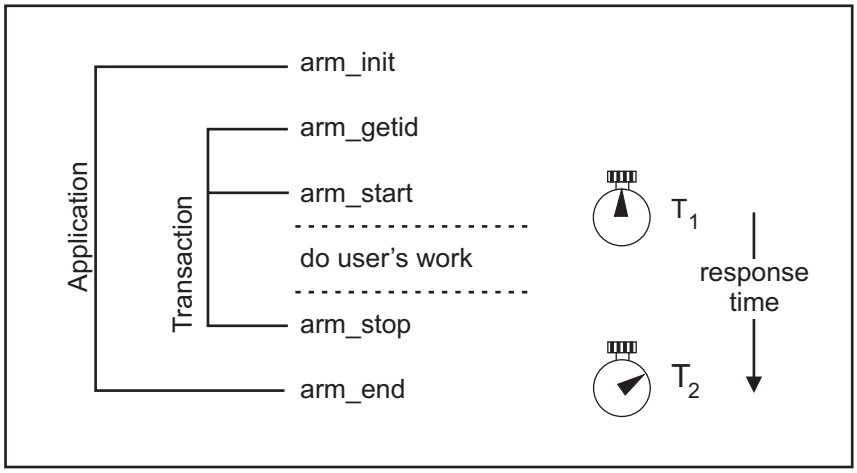


Figure 37. Relationship between ARM calls

The benefit of this approach is that you can place the calls that start and stop the response time clock in exactly the parts of the application that you want to measure. Instrumenting is done by defining individual applications and transactions within a program, and then placing the ARM API calls at the start of the transaction and at the end of the transaction.

How data is collected

Response time collection is performed by the management agent. For the product to collect response times for an application, the application must make a call to the ARM API on the computer where response time measurements are being made.

A management agent on which an ARM component is deployed can be installed on each of the computers within the path of a transaction. If matching start and stop calls are made for each component of the transaction on the machine on which the transaction is run, the transaction can be followed through the infrastructure.

Commercial applications with ARM instrumentation

You can use applications that are already instrumented. The following applications already contain ARM instrumentation:

- WebSphere Version 5.1 and 6
- Apache web server
- IBM HTTP server
- Microsoft Internet Information Server
- DB2 Version 8
- Siebel server Version 7.7
- Baan ERP based on porting set 7.0a or later

Overview of ARM instrumentation for Response Time Tracking

ARM instrumentation enables you to monitor the response time of requests or the work that an application is performing.

To monitor an application, it is important that the product receive notification of the following types of events:

- The start of a request serviced by the application
- The beginning and end of the components servicing a request within the application.

This information provides a basic understanding of the work the application is performing and monitors the overall response time of the applications requests and the sub-components that are used for servicing those requests.

The product defines an application as a program that is designed for a specific purpose. Examples of applications are:

- WebSphere Application Server
- Siebel Application Server
- Apache
- Microsoft Word

If you have multiple installations of one application, each installation is considered an instance of that application. This is true even if the installations are on different computers.

You must notify ARM about each application to be monitored, and then again about each specific instance of those applications. This enables the product to report statistics based on the application without regard to where it resides or what instance it is, or to report information based upon a specific instance of an application.

A *transaction* is the section of work that should be monitored and timed by the ARM engine. When a user hits a web page and receives a reply from the server, this is single transaction from a user perspective. From the server perspective, however, this transaction might be broken down into several units of work, each of which might be considered an ARM transaction.

In this product, an *edge transaction* is the first point in time that work being done to service a user request is monitored. Other units of work performed to process that edge transaction are referred to as *subtransactions*.

Planning your ARM instrumentation

Instrumenting an entire suite of applications can be a complex and expensive process. Completing this process in stages enables you to minimize the time spent on instrumentation while maximizing your benefit. This chapter gives a high-level plan for instrumenting your applications.

Step 1: Application-level instrumentation

Add basic entry and exit-level instrumentation to your applications. This is the least amount of instrumentation effort and allows you to monitor transactions and

report on each transaction response time. Adding entry and exit level instrumentation involves analyzing your application to find the following instrumentation points:

- All entry points to your application. An entry point is a place in the code that listens for external requests to perform work. Examples of entry points are servlets, web services, and remote EJBs.
- All exit points from your application. Exit points are points in the code that issue requests to external applications. Examples of exit points are JDBC or RMI calls.

When you find the entry and exit points for your application, add ARM calls to each of these points as follows:

- For entry points, add a start call when the outside request is received and a stop call when the application finishes processing the request.
- For exit points, add a start call when the application calls the external application and a stop call when the reply is received. In addition, issue a block call after each start call and an unblock call before each stop. This indicates that the application is waiting on an external resource before it can complete the required transaction.

Step 2: Tracing transactions across multiple applications

Next, enable passing of correlators whenever possible between your applications. For details on correlator use, see "Transaction correlation across multiple applications" on page 464.

Step 3: Continue to add correlation across multiple applications

Consider adding correlation to any non-standard communication that your application supports.

Step 4: Detailed Application Instrumentation

Go through any applications you have already instrumented and identify other points of interest to instrument. In this phase, you might also consider creating a setting that configures how much ARM instrumentation is enabled so that an administrator can decide how much ARM instrumentation to gather.

Comparing ARM versions

You can perform ARM instrumentation using ARM Version 2 or 4. Each version of ARM builds on the previous version, enabling you to perform increasingly more complex instrumentation.

The following table lists the supported versions of ARM and provides a brief summary of the functions included with each.

Table 81.

ARM Version	Languages supported	Function provided	Complexity
Version 2	C and C++	Response time Completion status Basic ARM API calls Transaction correlation that enables you to trace transaction steps on one computer or across multiple computers Additional metrics support that enables you to add identification information when running transactions	Simple (Optionally complex with customization)
Version 4	C, C++, and Java	Response time Completion status Tracing across machines Context data More product support A more mature API set that enables you to implement more functionality and performance More supported character sets for better internationalization (UTF8, UTF16, and so on)	Simple to Complex

Using ARM 2 API in simple applications

Applications must be instrumented for the ARM agents to measure response times.

An ARM agent is any agent that implements ARM and receives the program calls, such as the IBM Tivoli Composite Application Manager for Response Time Tracking management agent. This section addresses what it means to be instrumented, and shows how to instrument some sample applications written in C using ARM Version 2 calls.

In this section, each ARM API call is described and a code example is provided.

ARM API Calls

arm_init

Used to define an application. It must be made before any other ARM API calls related to that application. For example:

```
appl_handle = arm_init(application, appl_user_id, 0,0,0);
```

Where:

application Specifies the name for the application. This can be any name up to 128 bytes. Choose a name that will be meaningful in the reports.

appl_user_id This can be set to *, in which case the ID of the logged in user is pulled from the operating system.

The handle returned by this function is placed in the *appl_handle* variable, which is then used by the `arm_getid` function.

arm_getid

Used to define a transaction. The transaction must be a child of an application. For example:

```
getid_handle = arm_getid(appl_handle,
transaction,tran_detail,0,0,0);
```

Where:

transaction Specifies a name for the transaction. This can be any name up to 128 bytes. Choose a name that will be meaningful in the reports.

appl_handle The variable that was returned by the `arm_init` function.

tran_detail Specifies information for use on reports, such as the type of machine on which the transaction is running. Response Time Tracking product uses the `tran_detail` parameter to group transactions in boxes in the topology view. For example, for all EJB type transactions, J2EE uses the value EJB for the `tran_detail` parameter, which is then used by reporting to group the EJB transactions into a box in the topology view.

The handle returned by this function is placed in the *appl_handle* variable, which is then used by the `arm_getid` function.

arm_start

Call this function when the transaction starts running. It starts the response time clock. For example:

```
start_handle = arm_start(getid_handle,0,0,0);
```

Where:

getid_handle The variable that was returned by the `arm_getid` function.

The handle returned by this function is placed in the *start_handle* variable. The *start_handle* variable is used by the `arm_update` and `arm_stop` functions.

arm_update

This function can be used as a heartbeat, to check the progress of a longrunning transaction. It can be used as many times as you want between the arm_start and arm_stop calls. For example:

```
updaterc = arm_update(start_handle,0,0,0);
```

Where:

start_handle The variable that was returned by the `arm_start` function.

This function returns an error code rather than a handle. The error code is placed in the *updaterc* variable.

arm_stop

This call is made when the transaction stops running. It stops the response time clock. For example:

```
stoprc = arm_stop(start_handle,arm_status,0,0,0);
```

Where:

start_handle The variable that was returned by the `arm_start` function.

transaction _status Indicates the transaction status. Possible values are `ARM_GOOD`, `ARM_ABORT`, and `ARM_FAILED`. These can be used to separately record transactions that do not complete as expected.

This function returns an error code rather than a handle. The error code is placed in the *stoprc* variable.

arm_end

Make this call when the application ends. It cleans up the memory that the management agent has allocated for this application. For example:

```
endrc = arm_end(appl_handle,0,0,0);
```

Where:

appl_handle The variable that was returned by the `arm_init` function.

This function returns an error code rather than a handle. The error code is placed in the *endrc* variable.

Case-sensitivity

The ARM API calls are C functions and are case sensitive. All ARM API functions use lowercase. If you use uppercase:

- In C or C++, the compiler fails to resolve the external functions.
- In a language other than C, the call returns -1.

Sample

The bold text in this example is used to highlight the information that was entered to perform the ARM instrumentation.

```
/*************************************************************/
/* This is a simple demonstration program.          */
/* It tests the IBM TMTP agent for Windows.          */
/*************************************************************/
#define W32
#include <windows.h>
#include "arm.h"
#include <stdio.h>
#include <string.h>

void main()


/*************************************************************/
/* Define the variables.                */
/*************************************************************/

{ char application[128];
char transaction[128];
long appl_handle;
long getid_handle;
long start_handle;
char string[20];
long stoprc;
long endrc;

/*************************************************************/
/* Ask for application name and make arm_init call.      */
/*************************************************************/
```

```
printf("\nThis program generates transactions for TMTP.\n\n");
printf("Please enter the name of your application:\n");
scanf("%s", &application);
appl_handle = arm_init(application,"*",0,0,0);

/*************************************************************/
/* Ask for transaction name and make arm_getid call.      */
/*************************************************************/

printf("\nPlease enter the name of your transaction:\n");
scanf("%s",&transaction);
getid_handle = arm_getid(appl_handle,transaction,"ARM",0,0,0);

/*************************************************************/
/* Start transaction and make arm_start call.        */
/*************************************************************/

start_handle = arm_start(getid_handle,0,0,0);
printf("\nTransaction started...\n");
printf("Type some characters and press ENTER when\n");
printf("you want to stop the transaction.\n");
scanf("%s",&string);

/*************************************************************/
/* Stop transaction and make arm_stop call.          */
/*************************************************************/

printf("\n\nTransaction stopped.\n\n");
stoprc = arm_stop(start_handle,ARM_GOOD,0,0,0);

/*************************************************************/
/* End application by making an arm_end call.         */
/*************************************************************/

endrc = arm_end(appl_handle,0,0,0);
}

/*************************************************************/
/* End of program.                      */
/*************************************************************/
```

Returns from API calls

Each ARM function returns a 32-bit signed integer.

The following functions return handles that indicate status or are used on subsequent calls:

- arm_init
- arm_getid
- arm_start

The following functions return integers that indicate status:

- arm_update
- arm_stop
- arm_end

Handles and return codes

Declare integer variables in your program to hold the handles and return codes from the ARM API calls. The handles cannot be manipulated as integer variables. The handle has meaning only to the ARM agent and cannot be modified.

A negative return code always indicates that the function was unsuccessful. For calls that return handles, a positive integer indicates success. For calls that return status, zero (0) indicates success.

Table 82 gives the possible return codes for failure.

Table 82. ARM return codes

Return code	Explanation
-1	An invalid argument was specified. This usually means that a mistake was made in the syntax of the call. For example, too many or too few arguments might have been included on the call. In C or C++, these errors are caught by the linker when it resolves the external references in the shared library that is shipped with the agent. In other languages, there is no link step and the errors occur when the ARM API call is made.
-2	The ARM agent is not running. This error usually occurs because the ARM agent has not been started on the machine where the ARM API call is made. The agent must be installed and running.

Nested and overlapping transactions

A management agent with ARM monitoring component returns a handle in response to each ARM API call. This enables you to nest or overlap transactions. Figure 38 on page 456 is an example of overlapping transactions in application code:

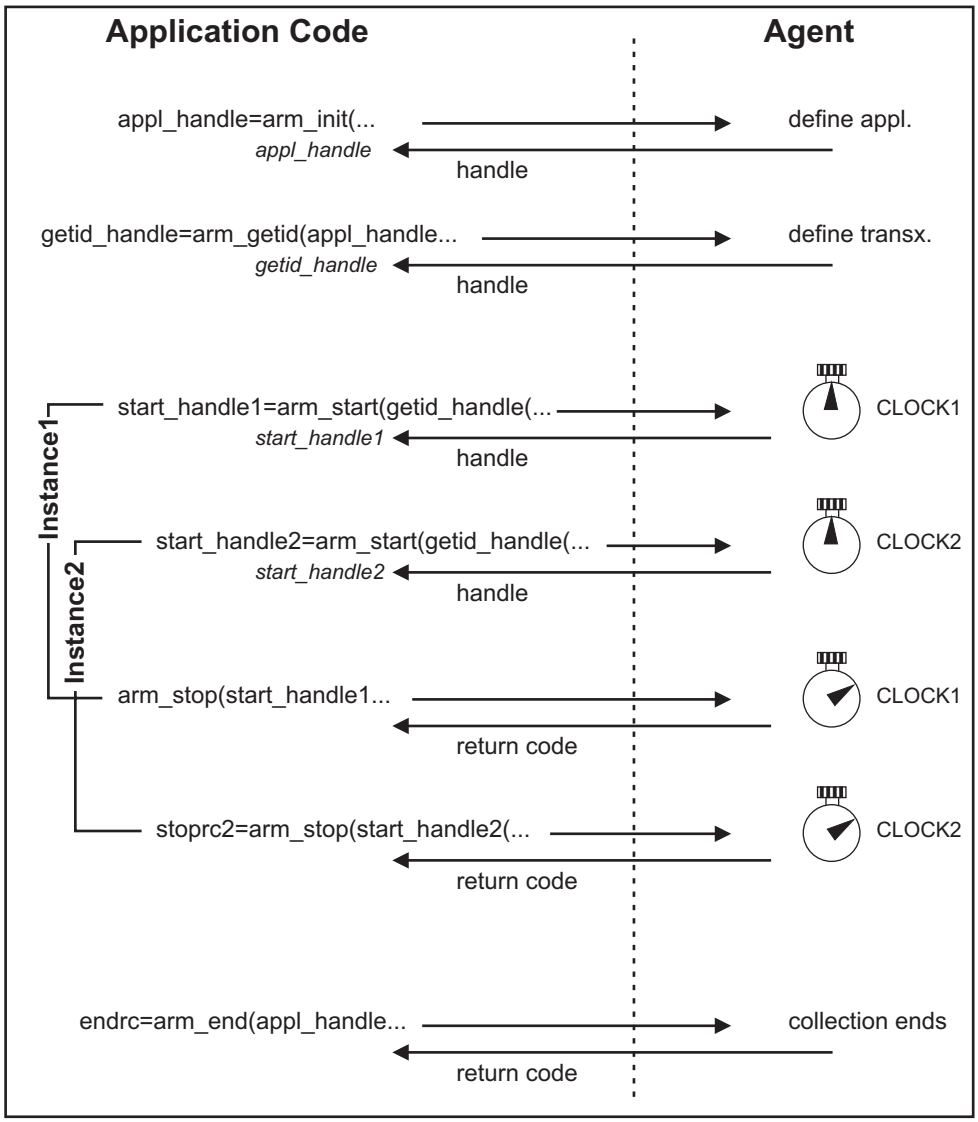


Figure 38. Example of overlapping transactions

Using ARM API calls efficiently

There are two API function calls in the ARM Version 2 API that are associated with registration, the arm_init function call and the arm_getid function call.

These API calls normally have a higher overhead than the corresponding arm_start function call and the arm_stop function call. These function calls should be performed once when the application starts, for the arm_init function, and once the first time the transaction is run, for the arm_getid function.

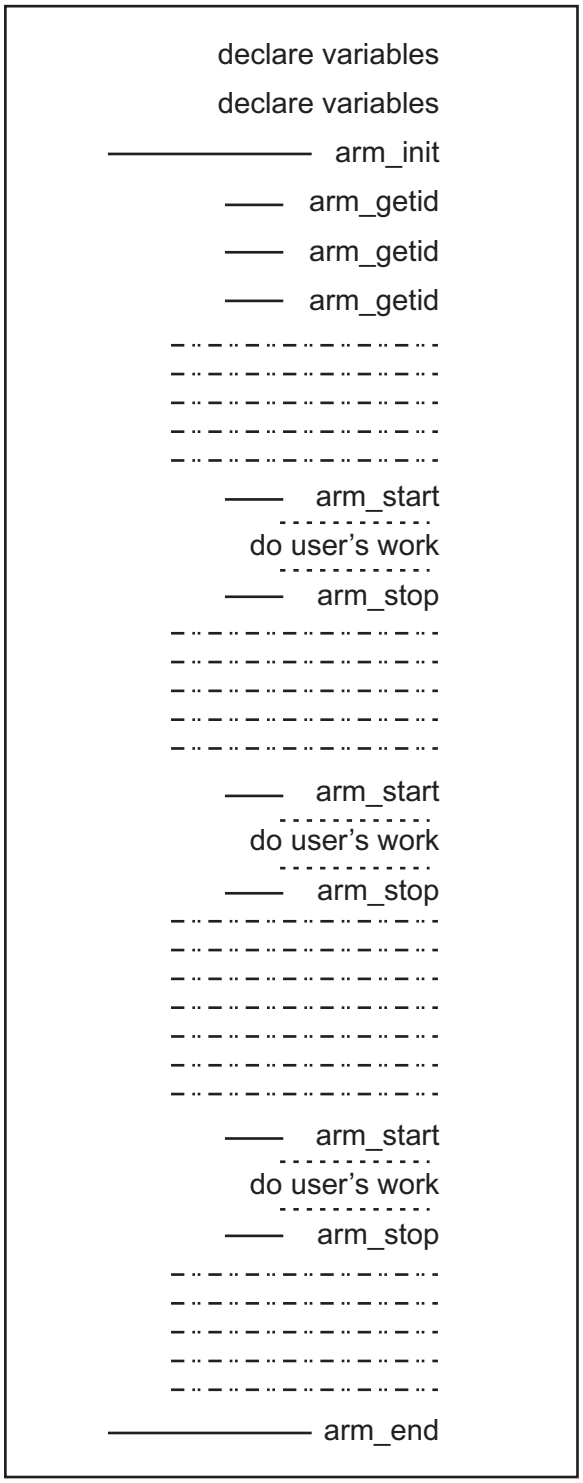


Figure 39. API calls

Using the completion status on the arm_stop call

Provide transaction status to the product by passing one of the following parameters to the arm_stop function:

Value	Explanation
ARM_GOOD	Indicates the transaction completed successfully.
ARM_FAIL	Can be used to create alarms if the transaction does not complete as expected.
ARM_ABORT	Can be used to create alarms if the transaction does not complete as expected.

The product separately reports transactions that complete in each of the three statuses.

Compiling with ARM 2

Once you have placed the ARM API calls in source file, you are ready to compile.

The compile process for an ARM-instrumented application is shown in Figure 40. You do not need to have an ARM agent installed or active on the machine where you are compiling.

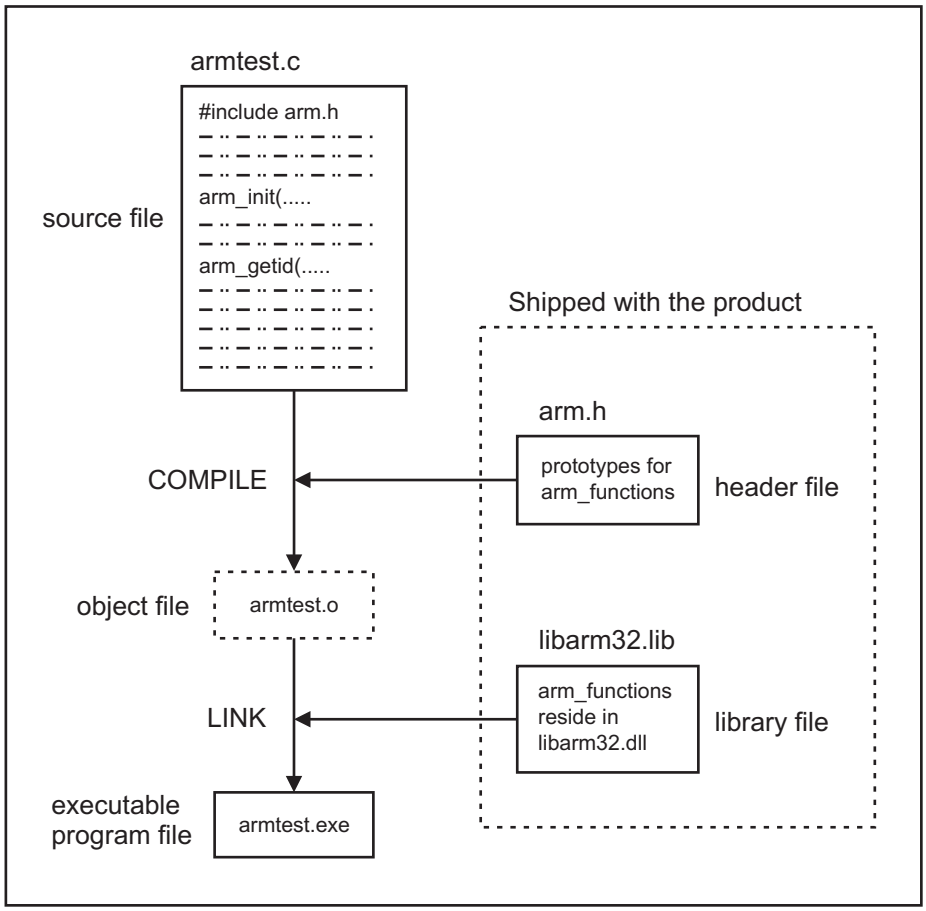


Figure 40. ARM-instrumented application compile process

Before compiling, check that the following requirements are met:

1. That the required header and library files are present on the machine. The ARM header and library files provide the prototypes for the ARM functions

and resolve those functions during the link step. You need the header and library files appropriate for the operating system platform on which the application will run.

2. That the directory containing the header and library files has been specified in the compiler's include and in the link directory paths.
3. That you have specified the correct library file to the linker. Each operating system platform requires a different library file.

Note: The ARM header libraries define the **typedef** parameter that can conflict with parameters defined by the operating system's header files. Maintain the definition of the ARM library so that ARM correlators are generated correctly.

Using correlation to link activities or applications

The ability to model complex, multi-step transactions is one of the strengths of ARM instrumentation and is accomplished with correlators.

A *correlator* is a data structure that contains fields unique to the transaction instance for which it was generated. *Correlator passing*, or *correlation*, occurs when you send the correlator between related transactions. Correlator passing is essential to a well-instrumented application and allows you to:

- Break a transaction into its component parts, enabling you to see each component's contribution to the total response time of the application.
- Reconstruct the path of the overall transaction and see the relationship between the various component transactions. It is usually possible for a transaction to follow more than one path.

To trace a multi-step transaction across its subtransactions, request a correlator from the management agent on the **arm_start** function call. The instrumented code must then pass the correlator on to the next step in the transaction, which can be on the same computer, or another computer, and it must then be passed in on the **arm_start** function call that is made in that subtransaction.

Figure 41 on page 460 shows the correlator being sent between transactions.

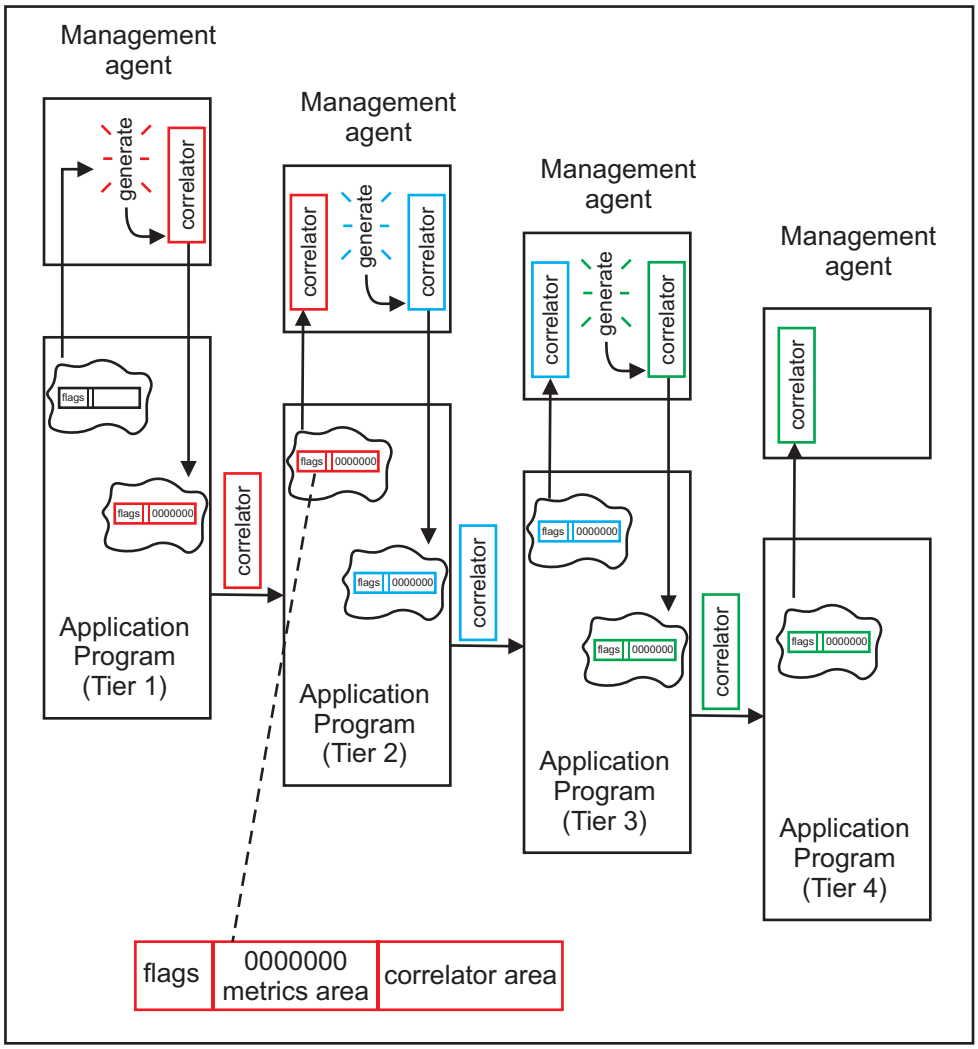


Figure 41. Correlator passing

Figure 42 on page 461 shows the topology of a WebSphere Application Server transaction and enables you to see the relationship between the various component transactions:

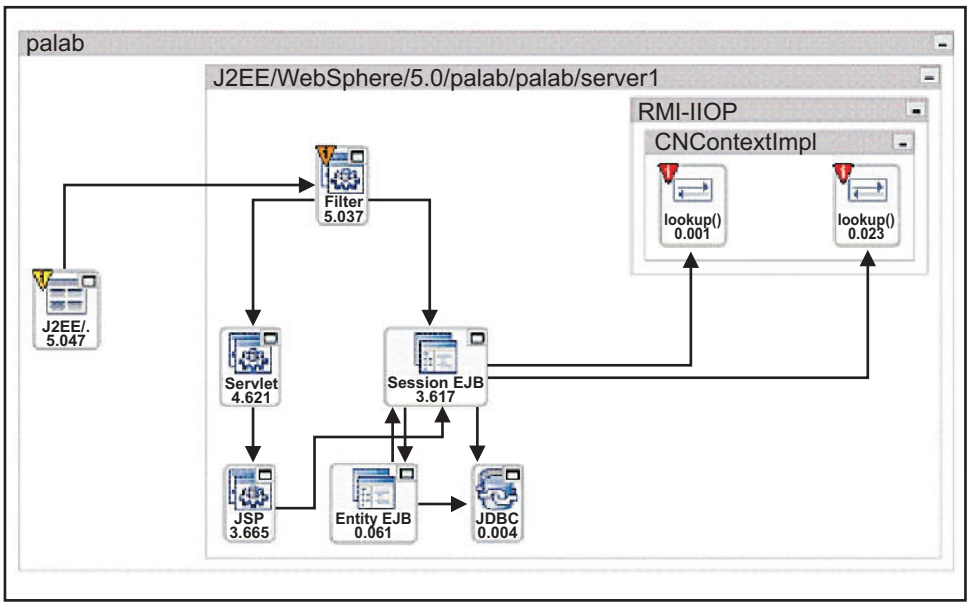


Figure 42. Example Topology

Single application correlation

Correlating across a single application is the simplest way to use correlation. This example implements correlation across a single application and mimics a method level trace of the application activity.

Suppose an application received a request to look up a user name, given a user ID. The entry into the application might be an **`HttpServlet`** called `LookupUserNameServlet` and the flow might be as shown in Figure 43.

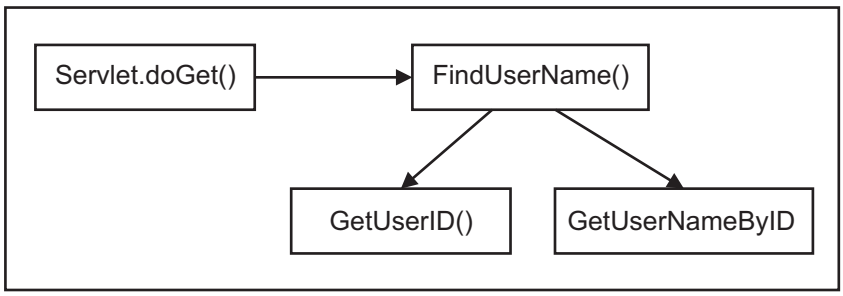


Figure 43. Lookup request

To do method-level tracing with correlators, issue a stop and start call for each method as you enter and exit it. In addition, pass the `doGet` correlator to the `FindUserName` method. Then generate a `FindUserName` correlator and pass it into `GetUserID` and `GetUserNameByID()`. Because `GetUserID` and `GetUserNameByID` do not call any methods, their correlators are not passed to any other methods.

```
public class LookupUserNameServlet extends HttpServlet {
String FindUserName () {
 // issue arm start call
 Integer userID;

 // lookup the userID
 userID = GetUserID();

 // issue arm stop call
```

```
 return GetUserNameByID(userID)

}
Integer GetUserID() {
 // issue arm start call
 // find the user id
 // issue arm stop call
 return rv;

}

String GetUserNameByID(Integer userid) {
 // issue arm start call
 // find the username
 // issue an arm stop call
 return username;

}

void doGet(HttpServletRequest req, HttpServletResponse resp) {

 // issue arm start call
 // lookup the user using findUserName
 FindUserName();
 // issue Arm stop call

}

}
```

To avoid modifying each method's signature, create a thread local stack. In Java this is done using the `java.lang.ThreadLocal` and `java.util.Stack` classes. This is best done by creating an `ArmUtility` class to handle correlators. For example:

```
public class ArmUtilities {
 private static ThreadLocal CorrelatorStack = new ThreadLocal() {
  protected synchronized Object initialValue() {
   return new Stack();
 }
};

public ArmCorrelator getCorrelator() {
 Stack corStack = (Stack) CorrelatorStack.get();
 // get a copy of the parent correlator.
 ArmCorrelator parentCorrelator = corStack.peek();
 return parentCorrelator;
}

public void pushCorrelator(ArmCorrelator currentCorrelator) {
 // get this threads stack
 Stack corStack = (Stack) CorrelatorStack.get();
 // push the current correlator onto the top of the stack
 corStack.push(currentCorrelator);
 // update the thread local variable
 CorrelatorStack.set(corStack);
}

public void popCorrelator() {
 // get this threads stack
 Stack corStack = (Stack) CorrelatorStack.get();
 // pop the current correlator off the top of the stack
 corStack.pop();
 // update the thread local variable
 CorrelatorStack.set(corStack);
}
}
```

The servlet ARM instrumentation code to handle the correlator is as follows:

```
// This class supports correlation
public class LookupUserNameServlet extends HttpServlet {

String FindUserName () {
 // Get the parent correlator
 ArmCorrelator parentCorrelator = ArmUtilities.getCorrelator();
 // issue arm start call passing in the parent correlator
 // Now save the new correlator generated
 ArmUtilities.pushCorrelator(currentCorrelator);

 Integer userID;

 // lookup the userID
 userID = GetUserID();

 // Pop the correlator off the stack since this method is done.
 ArmUtilities.popCorrelator();
 // issue arm stop call
 return GetUserNameByID(userID)
}

Integer GetUserID() {
 // Get the parent correlator
 ArmCorrelator parentCorrelator = ArmUtilities.getCorrelator();
 // issue arm start call passing in the parent correlator
 // Now save the new correlator generated
 ArmUtilities.pushCorrelator(currentCorrelator);

 // find the user id

 // Pop the correlator off the stack since this method is done.
 ArmUtilities.popCorrelator();
 // issue arm stop call
 return userid;
}

String GetUserNameByID(Integer userid) {
 // Get the parent correlator
 ArmCorrelator parentCorrelator = ArmUtilities.getCorrelator();
 // issue arm start call passing in the parent correlator
 // Now save the new correlator generated
 ArmUtilities.pushCorrelator(currentCorrelator);

 // find the username

 // Pop the correlator off the stack since this method is done.
 ArmUtilities.popCorrelator();
 // issue arm stop call
 return username;
}

void doGet(HttpServletRequest req, HttpServletResponse resp) {
 // Get the parent correlator
 ArmCorrelator parentCorrelator = ArmUtilities.getCorrelator();
 // issue arm start call passing in the parent correlator
 // Now save the new correlator generated
 ArmUtilities.pushCorrelator(currentCorrelator);

 // lookup the user using findUserName
 FindUserName();

 // Pop the correlator off the stack since this method is done.
 ArmUtilities.popCorrelator();
 // issue arm stop call
}
}
```

In C or C++ this can be accomplished using a hash table where the key is the `threadID` and the value is a simple stack object.

Transaction correlation across multiple applications

Correlating across multiple applications allows you to track a transaction between different resources in your IT infrastructure in order to service user requests.

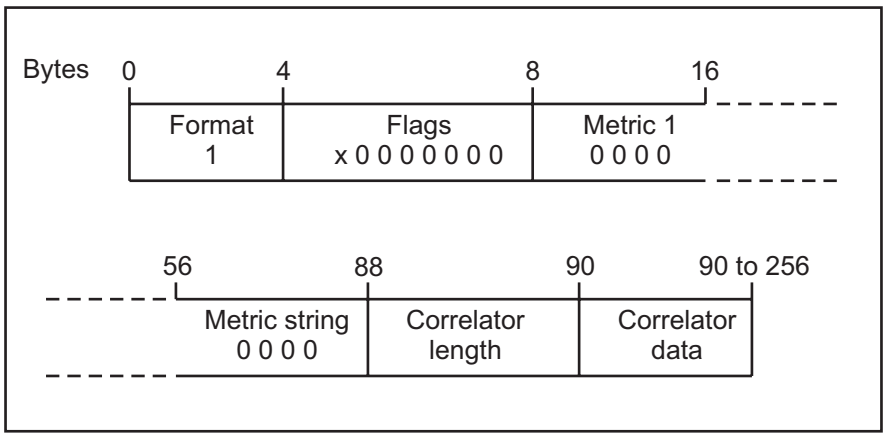
Additionally it allows you to see the performance dependencies between those resources and identify the problem resource when performance or availability problems occur.

The ARM group does not currently define a standard for passing correlators between applications. However, ITCAM for Transactions uses a common IBM mechanism for passing correlators between applications for common protocols. The following sections describe that mechanism.

The correlator data structure

A correlator is stored as part of the ARM Additional Data buffer. The Additional Data buffer must be set up by the application before the arm_start function call is made.

This buffer can contain data pertaining to ARM additional metrics and transaction correlation. We will only look at transaction correlation, in which case all the metric fields in the buffer can be zero (0).



Three parts of the buffer are relevant for transaction correlation:

- Flags

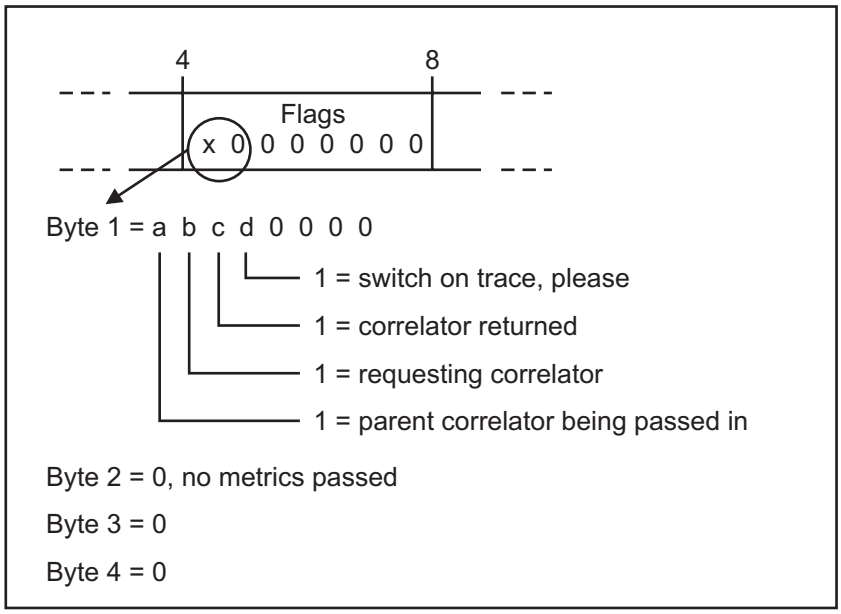
  Used to control correlator handling in the ARM agent.
- Correlator length

  Contains the length of the Correlator Data section, is variable, and can be up to 168 bytes for ARM 2 and up to 512 bytes for ARM 4. This is stored in the Big Endian format.
- Correlator data

  Contains the correlator.

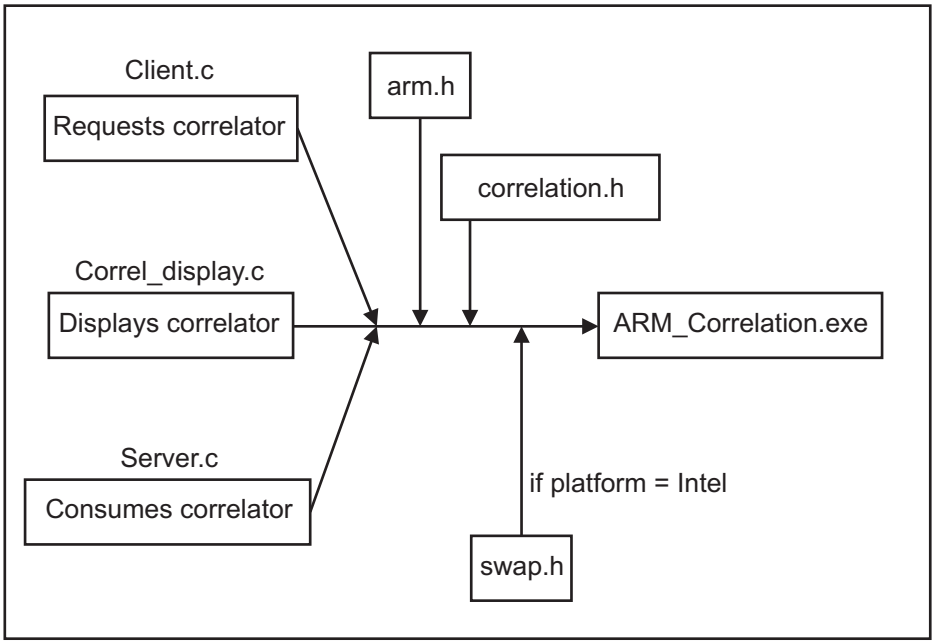
The flags section is used to control the behavior of correlator handling for the ARM agent; only the first four bits of the first byte are relevant for transaction correlation.

There are four bit-switches that can be used in correlation. The switches can be set by the application before making the **arm_start** function call, and by the ARM agent in response to the call.



Sample

A sample program is available, in source and compiled form, to test correlation functions. Refer to this sample while reading the sections about performing correlation. Download this program from InfoCenter. The sample program consists of several components, as shown below:



The client component makes an **arm_start** function call requesting a correlator. If a correlator is returned, it is displayed on the screen. Here is a sample of output:

```
Buffer flags section contains: 64
Client start call made: handle = 17
Now the buffer flags section contains: 32
----------------------- correlator-----------------------
correlator = |01|00|00 01|06 3e|00 02|00 00|00 0000 11|00
      00 00 01|00 04|ff 54 12 45|00 00
Length = 26
Format = 1
Flags = 0
Address format = 1
Vendor Id = 1598
Agent version = 2
Agent Instance = 0
Trans Instance = 17
Trans class Id = 1
Address length = 4
------------------------------------------------------------
```

Then the server component consumes the correlator.

Working with correlators in ARM 2

To use ARM transaction correlation you must do the following:

1. Generate a correlator when the arm_start function call of a parent transaction is made.
2. Pass the returned correlator to the next step so that it is available if an arm_start function call is made for a child transaction.
3. Pass the correlator in from the buffer, to be consumed by a child transaction, when an arm_start function call is made.

Generating a correlator in ARM 2

Generate a unique correlator for each transaction.

About this task

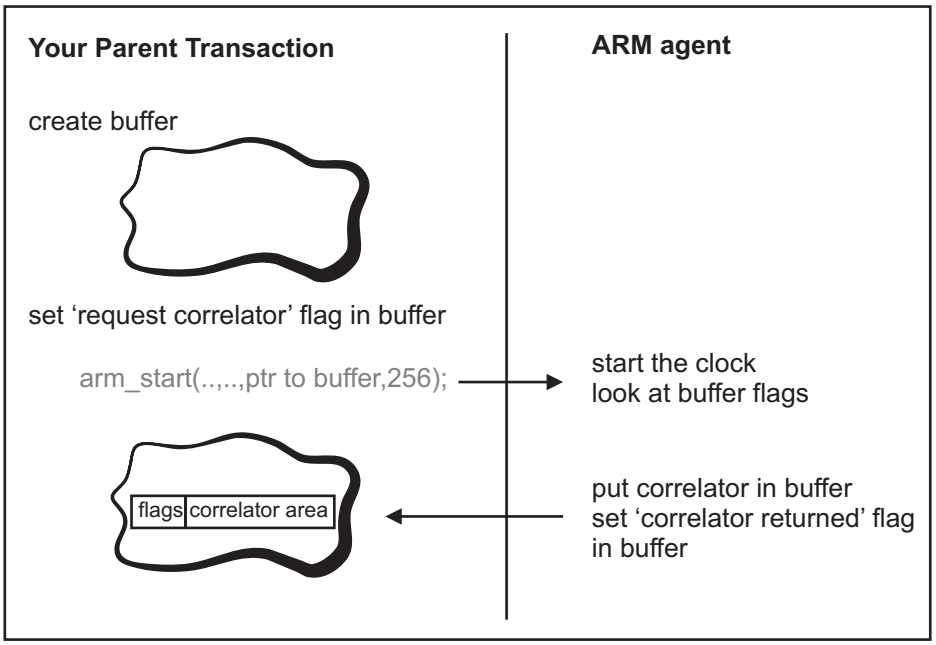
To generate a correlator:

Procedure

1. Create a buffer in the transaction.
2. Set the request correlator flag in the data buffer.
3. Make an arm_start function call that has parameters pointing to the buffer and setting the maximum correlator length for the requested correlator. The maximum correlator length is 168 bytes.

Results

The management agent will return a correlator, put it in the buffer, and set the correlator returned flag.



Passing a correlator

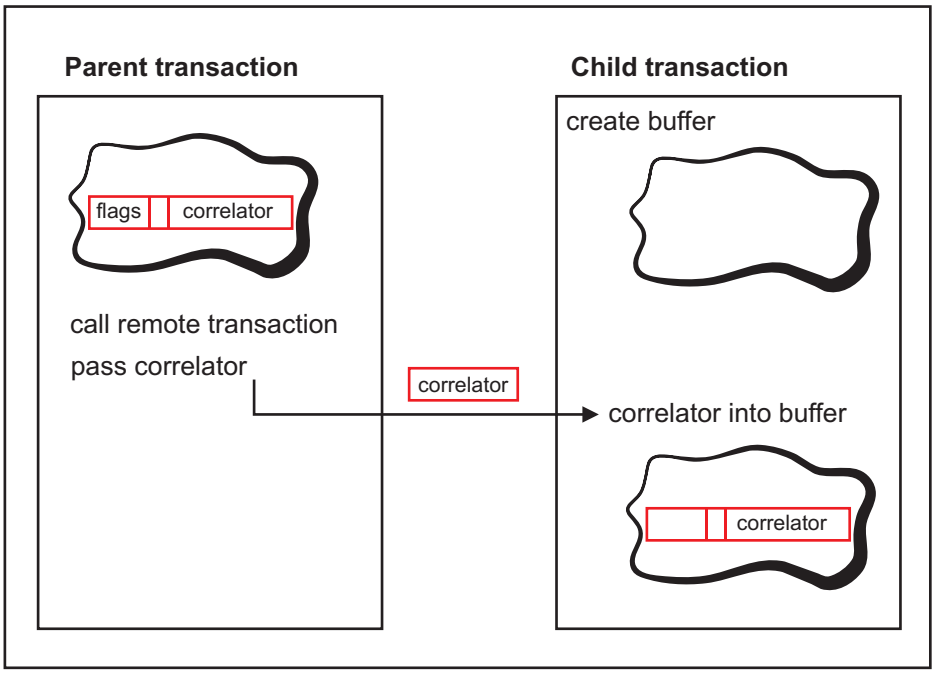
After the correlator is generated, the instrumented code must then pass the correlator on to the next step in the transaction, which can be on the same computer, or another computer.

About this task

This correlator must be stored in a buffer that can be read when an arm_start function call is made for one of its child transactions and is requesting a correlator.

It might also be beneficial to pass correlators to external applications (or processes if your application runs in multiple processes). This enables the ARM engine servicing the other application to associate work it performs in that application with the work done by the external application to build an inter-application transaction tree view.

To generate a correlator:



Consuming a correlator in ARM 2

When the child transaction is run, a flag must be set in the buffer to request that a parent correlator be passed in.

About this task

To set the flag:

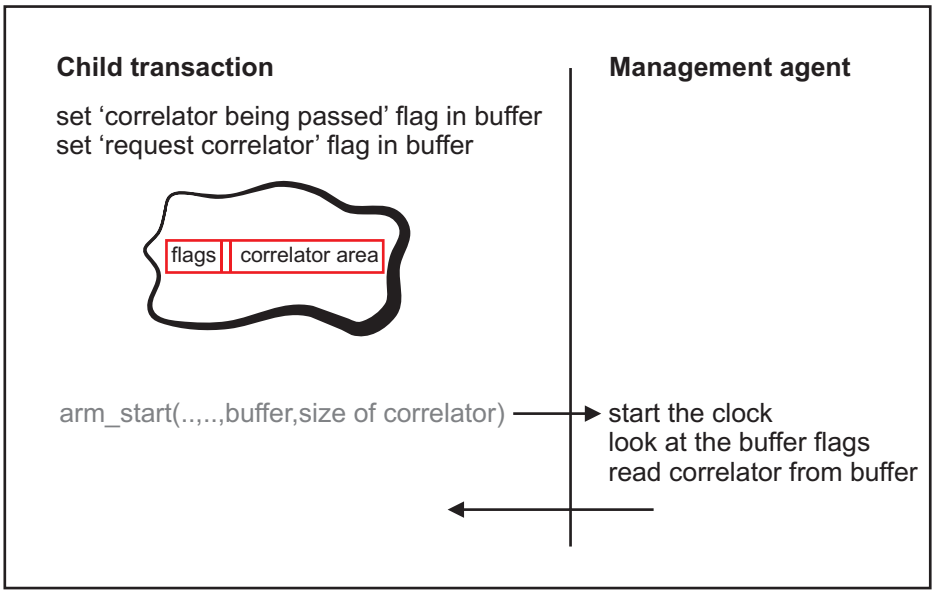
Procedure

1. Set the flag in the buffer to pass in the parent correlator.
2. If this transaction will call another transaction, a grandchild transaction, set the request correlator flag in the buffer.

   Note: A new correlator must be requested at each step.
3. Make the arm_start function call with parameters pointing to the buffer and specifying the length of the correlator that is being passed in.

Results

The management agent will consume the parent correlator that is passed in. If a new correlator was requested, the management agent will also return a new correlator, put it in the buffer, and set the correlator returned flag.



Example: Tracing across multiple applications

Correlating across multiple applications allows you to initialize the thread local stack of a downstream application with the last correlator generated, top of the stack, from the upstream application.

This is accomplished by performing the following generic code on every exit point of an application:

- Application 1:

```
exit point:
// get the correlator off the top of the stack
ArmUtilities.getCorrelator();

// pass the correlator using the protocol specific code.
```

- Application 2:

```
// receive the correlator from application 1 using the protocol specific
// code

// initialize the stack with the correlator received.
ArmUtilities.pushCorrelator();
```

Note: When a correlator is received, pop the extra correlator that you received from the top of the stack. For entry points it might be useful to define a `cleanStack()` method that empties the stack instead of the standard `popCorrelator()` method call when exiting the method.

Tracing a transaction and writing context data

ARM Version 2 provides transaction correlation and additional metrics in addition to the Version 1 capabilities.

Registering your application: ARM Version 2

In ARM Version 2, you must register your application before it can be monitored. Registration is done using the arm_init function.

When you register an application, you are informing ARM that it should be monitored. Typically an application registers itself each time it is started and passes contextual information for identification. When the application ends, it notifies the ARM agent that it is no longer running and that any resources allocated on the local machine can be freed up for other uses.

The application name passed to the arm_init function should be the same every time the application is started and for every instance of this application. This enables you to identify the application and the user that runs the application. The return value from this call needs to be saved for later use and is a great candidate for encapsulation into a singleton object for the application.

Using C or C++

```
* Obtain/create a handle for the given Application/User pair.
*
*appl_name ASCIIZ pointer to the application name. Must not be
* NULL or an empty string.
*appl_user_id ASCIIZ pointer to the user name. If NULL, will use
* an empty string as the name. If * will get the currently
* running user from the base operating system.
*flags Reserved parameter.
*data Reserver parameter. Not used.
*data_size Reserver parameter. Not used.
*
*return Value > 0 containing a handle to the application/user
* pair, otherwise an error code (<= 0).
*/

arm_int32_t ARM_API arm_init(char *appl_name,
             char *appl_user_id,
             arm_int32_t flags,
             char *data,
             arm_int32_t data_size);
int applicationID=-1;
applicationID = arm_init("WebSphere","*",0,null,0);

/**
*Registers an application Identified as "Websphere" running as
*the logged in user.
*/
```

Registering your transaction: ARM Version 2

Register a transaction to inform ARM about a type of transaction that will be run. When you register a transaction, you are providing contextual information that is used by the ARM engine to classify that transaction.

Transactions are usually registered once when the application is started, or when a class is loaded in Java using a static initializer for that class. Information that changes from instance to instance is described at registration and then is passed at the time the transaction starts.

Transactions must be organized and formatted so they can be easily understood by the product collecting the ARM data and the user who is viewing that data. For these reasons it is important that you plan carefully when creating transaction names.

For edge transactions, use more descriptive names from a business perspective. Examples of descriptive names are:

- URLs
- Script names and useful parameters
- The action being performed and the object on which it is being performed. For example: `MicrosoftWord.SpellCheck("ArmInstrumentationGuide");`

Use this type of naming convention on subtransactions if you are using an engine that decodes information and performs work based upon the values passed to it. For example, a subtransaction named `runScript(String)` is not as useful as `runScript(actualscriptname)`.

Note: In the following examples a transaction is registered for an object called Foo and method named bar with signature (String) or Foo.bar(String).

Using C or C++:

```
/**
* Obtain/create a transaction class ID for the input transaction name
* associated with the input Application/User handle.
* Each transaction name associated with an application/user handle is
* unique. The same transaction name may exist for 2 or more
* application/user handles but they will each have a unique transaction
* class ID.
* *
appl_id Application/user handle previously returned by a
* successful call to arm_init().
*tran_name ASCIIZ pointer to the transaction name.
*tran_detail ASCIIZ pointer to the transaction detail. Must not be NULL.
*flags Not used.
*data Must point to an arm_user_data101_t element (see arm.h).
* Contains the metrics meta data for all metrics used by this
* transaction. This metric data is passed in arm_start, arm_update,
* arm_stop, etc.
* The format element must be ARM_Format101.
*data_size Number of bytes pointed to by data.
* *
return Value >
0
containing a
handle to the application/user
* pair, otherwise an error code (<= 0).
*/

arm_int32_t ARM_API arm_getid(arm_int32_t appl_id,
              char* tran_name,
              char* tran_detail,
              arm_int32_t flags,
              char* data,
              arm_int32_t data_size);
int transactionID;
applicationID = arm_init("WebSphere/Cell/Node/Server","*",0,null,0);
transactionID = arm_getid(applicationID,"Foo.bar(String);","",0,NULL,0);
```

Starting and stopping a transaction: ARM Version 2

This section describes how to use the handles returned by the ARM functions to run transactions that are monitored by the application.

Note: Without the use of correlators, products using the ARM are unable to establish relationships between transactions and their subtransactions. The examples in this section do not make use of correlators.

Using C or C++:

```
* Mark the beginning of execution of a transaction. Each time a
* transaction runs, it is called a transaction instance.
* Parameters:
* tran_id: the unique identifier assigned to the transaction
* class
* flags: reserved = 0
* data: a pointer to a buffer with additional data that can
* optionally be passed
* data_size: length in bytes of the buffer pointed to by data
* Return value:
* start_handle: the unique transaction handle assigned to this
* instance of a transaction
**/
arm_int32_t arm_start( arm_int32_t tran_id,
arm_int32_t flags,
char *data,
arm_int32_t data_size);
/**
* Mark the completion of a transaction instance that was started with a;
* call to arm_start().
*
*start_handle The handle to a successfully started transaction
* i.e. return value from arm_start()).
*tranStatus The status of the transaction. Valid values are the
* values of arm_tran_status_e (see arm.h):
* ARM_GOOD - Transaction was successful.
* ARM_ABORT - Transaction was aborted before it could finish.
* ARM_ABORT - Transaction was aborted before it could finish.
*flags Reserved = 0.
*data Pointer to a arm_user_data1_t structure containing
* metric information for the transaction. Can be NULL
* if no metric information is to be passed.
*data_size Number of bytes pointed to by data. Should be 0 if
* data is NULL.
*
*return 0 if the transaction instance is successfully
* stopped, an
* error code otherwise (< 0).
*/

arm_int32_t ARM_API arm_stop(arm_int32_t start_handle,
             arm_int32_t tranStatus,
             arm_int32_t flags,
             char *data,
             arm_int32_t data_size);
int applicationID;
int transactionID;
int startHandle;

applicationID = arm_init("WebSphere/Cell/Node/Server","*",0,null,0);
transactionID = arm_getid(applicationID,"Foo.bar(String);","",0,NULL,0);

/**
*
*Appication logic
```

```
*
*/

startHandle = arm_start(transactionID,0,NULL,0);
arm_stop(startHandle,ARM_GOOD,0,NULL,0);
```

Ending applications and transactions: ARM Version 2

This section describes how to use the ARM end function call to clean up the memory that the management agent has allocated for this application.

Using C or C++:

```
/** Mark the end of an application
*
*
*appl_id Application/user handle previously returned by a
* successful call to arm_init()
* flags Reserved. Set this to 0.
* data Reserved. A NULL pointer. Set this to 0.
* data_size Reserved. Set this to 0.
*
*return 0 if the call is successful
*
*/

arm_int32_t ARM_API arm_end(arm_int32_t appl_id,
             arm_int32_t flags,
             char* data,
             arm_int32_t data_size);
endrc = arm_end(appl_id,0,0,0);
```

Working with Java applications: ARM Version 2

This product provides a Java Native Interface (JNI) implementation for ARM Version 2, so that ARM Version 2 calls can be made from Java applications. The JNI will translate them into C calls using the ARM Version 2 standard. This is less complex, but also less powerful than the Java bindings that are included in ARM Version 4.

Note: If you use ARM Version 4 Java bindings, the product uses a JNI implementation to convert them to C as well.

Using the JNI to make calls to a native method potentially compromises the Java environment as follows:

- Java applications could be lost since the native methods are operating system dependent. The product provides the JNI implementation for all platforms that are supported as management agents.
- There could be a security risk, since it might permit access to host operating system services. This product handles this by requiring that the management agent be installed on the machine where calls are being made.

To give your Java application access to the JNI, add the appropriate ARM JAR file to the class path on the Management Agent machine where your ARM-instrumented application will run. For ARM Version 2 calls, this is:

Windows

```
%MA\lib\armjni.jar
```

UNIX

```
$MA/lib/armjni.jar
```

Making ARM API calls from your Java classes

You can make ARM Version 2 calls from any of your Java Classes by doing the following:

1. Code an import statement for the class ARMClass.
2. Make the ARM API calls, using the standard C calling convention, with the following modifications:
   - For each of the ARM API functions, add the ARMClass class to the function name.
   - All of the ARM API calls have an argument called data which must be written as zero (0). This argument is a char pointer and in Java, a char pointer must be a string. For example:

     ```
     int start_handle= ARMClass.arm_start(getid_handle,0,"0",0);
     ```

Java examples for performing ARM Version 2 instrumentation

This section contains Java examples corresponding to the steps involved in preforming ARM Version 2 instrumentation.

Registering your application using Java

```
package com.tivoli.tapm.armjni;
/** Tells ARM you have an application/user pair that wishes to make ARM
* calls.
* @param appName Name of the application (128 ASCIIZ)
* @param userId user name (128 ASCIIZ) Specifying "*" for the username
* tells ARM engine to use the user ID that the process is
* currently running under.
* @return application ID to use for future armGetId calls.
**/

public static native int armInit(String applName, String userId);
int appl_id;
com.tivoli.tapm.armjni.ArmJni.armInit("Websphere","*");

/**
*Registers an application Identified as "Websphere" running as whatever
*user on the local machine started the engine.
*/
```

Registering your transaction using Java:

```
package com.tivoli.tapm.armjni;
/** Tells ARM you have a unique class of transaction you wish to monitor
* for your application
* @param appId returned by a previous armInit call
* @param tranName name of the transaction (2k ASCIIZ)
* @param tranDetail Description of the transcation (128 ASCIIZ)
* @param metricInfo array of metric info description objects.
* @return transClassID a unique ID used to identify this transaction.
**/

public static native int armGetId(int applId, String tranName,
  String tranDetail,MetricInfo[] metricInfoSet);

int appl_id;
int tran_id;
appl_id = ArmJni.armInit("Websphere/Cell/Node/Server","*");
tran_id = ArmJni.armGetId(appl_id,"Foo.bar(String);","",null);
```

Starting and stopping a transaction using Java:

```
package com.tivoli.tapm.armjni;
class ArmJni;
/**
* arm_start indicates that an instance of a transaction has begun
* execution.
* Contrast this with arm_getid, which defines the name of the
* transaction class during initialization, but doesn't actually indicate
* that a transaction is executing. The identifier returned on the
* arm_getid call is passed as a parameter on arm_start so an agent knows
* which type of transaction is starting. There can be any number of
* instances of the same transaction class executing simultaneously on the
* same system.
* To identify each one, the return code from arm_start is a unique handle
* generated by the agent. This handle is unique within a system
* across all instances of all transactions of all applications.
* The transaction instance is the fundamental entity that the ARM API
* measures. The parent correlator is passed within the data paramater,
* and the new correlator is returned in place of the parent correlator.
* @param transclassid returned from a previous ARM getid call
* @param MetricData metric data as described in a previous ARM getid call.
* @param correlator input correlator. Must not be null.
**/
public static int armStartWithCorrelator(int tranId,
MetricData[] metricDataSet,
Correlator correlator ) ;
/**
* arm_stop indicates that an instance of a transaction has completed.
* The handle returned on the arm_start call is passed as a parameter
* arm_stop can be issued from a different thread or process
* from which the arm_start was issued.
* @param transaction id return from a previous ARM start call.
* @param status The transaction status is also passed, and there
* are three possible values.
* ARM_GOOD - the transaction completed normally and the intended service
* was performed.
* ARM_ERROR - the transaction completed with an error (so the intended
* service was not performed).
* ARM_ABORT - the transaction was unable to complete. An example
* of a reason for an abort would be a
* time-out error from a communications stack.
* @return 0 on success.
**/

public static int armStop(int tranId, int status,
MetricData[] metricDataSet)

int appl_id;
int tran_id;
int startHandle;

appl_id = ArmJni.armInit("Websphere/Cell/Node/Server","*");
tran_id = ArmJni.armGetId(appl_id,"Foo.bar(String);","",null);

startHandle = ArmJni.armStartWithCorrelator(tran_id,null,null);
ArmJni.armStop(startHandle,ArmJni.ARM_GOOD,null);
```

Adding additional metrics for a transaction instance: ARM Version 2

You can further supplement this basic information by adding requirements to the instrumentation of the application. For example, with the notification about the start of an overall request serviced by the application, pass in additional metrics about the type of request and how it is related to the business objects the application is servicing.

This provides information on the type of requests the application is servicing and enables the user to:

- Know how long each type of request is taking.
- Manage the types of requests differently based upon configurable management policies.
- Place thresholds upon specific requests based upon different Service Level Agreements.
- Take corrective actions in the event that a request is slowing or not working at all.

ARM Version 4: for C, C#, and Java applications

This section contains information that enables you to instrument ARM Version 4 for Response Time Tracking.

To instrument an application, follow these steps:

1. (First time only) Set your Java properties.
2. Register your application.
3. Create an application instance.
4. Register your transaction.
5. Start and stop your transaction.
6. (Optional) Correlate transactions.
7. (Optional) Pass correlators.

Setting C# Environment Variables to use ARM Version 4

The C# specification is currently under review by the OpenGroup and until accepted it should be considered properietary to IBM's implementation.The C# implementation was essentially a port of the Open Groups Java API to C#.

Add the following environment variables before starting your .NET program. These properties define the classes that implement the .NET ARM Version 4 interface defined in the ARM Version 4 standards documentation.

```
set arm40.ArmTransactionFactory=IBM.arm40.transaction.Arm40TransactionFactory
set arm40.ArmTranReportFactory=IBM.arm40.metric.Arm40MetricFactory
set arm40.ArmMetricFactory=IBM.arm40.tranreport.Arm40TranReportFactory
```

You can use these properties to initialize the factories using the following static initializer:

Setting Java properties to use ARM version 4

Add properties to define the classes that implement the ARM Version 4 interface defined in the ARM Version 4 standards documentation.

For more information on the ARM Version 4 standards, see Java technical standards documentation.

Add the following properties to the startup command for your java program or define them during startup using `System.setProperty(Name,Value)`:

```
-DArm40.ArmTransactionFactory=com.ibm.tivoli.tt.ArmTransactionFactory
-DArm40.ArmTranReportFactory=com.ibm.tivoli.tt.ArmTransactionFactory
-DArm40.ArmMetricFactory=com.ibm.tivoli.tt.ArmTransactionFactory
```

You can use these properties to initialize the factories using the following static initializer:

```
/** initializes the factories.*/
 static {
  try {
   Properties p = System.getProperties();
   String keyTranFactoryClass =
    ArmTransactionFactory.propertyKey;
   String tranFactoryName =
    p.getProperty(keyTranFactoryClass);
   if ( tranFactoryName == null ) {
    System.err.println("Could't getProperty " +
     ArmTransactionFactory.propertyKey);
   } else {
    Class tranFactoryClass =
     Class.forName(tranFactoryName,true,
     ClassLoader.getSystemClassLoader());
    tranFactory = (ArmTransactionFactory)
    tranFactoryClass.newInstance();
   }

   String keyTranReportFactoryClass =
    ArmTranReportFactory.propertyKey;

   String reportFactoryName =
    p.getProperty(keyTranReportFactoryClass);

   if ( reportFactoryName == null ) {
    System.err.println("Could't getProperty " +
     ArmTranReportFactory.propertyKey);
   } else {
    Class reportFactoryClass =
     Class.forName(reportFactoryName,true,
     ClassLoader.getSystemClassLoader());
    reportFactory = (ArmTranReportFactory)
    reportFactoryClass.newInstance();
   }

   String keyMetricFactoryClass =
    ArmMetricFactory.propertyKey;

   String metricFactoryName =
    p.getProperty(keyMetricFactoryClass);

   if ( metricFactoryName == null ) {
    System.err.println("Couldn't getProperty " +
     ArmMetricFactory.propertyKey);
   } else {
    Class metricFactoryClass =
     Class.forName(metricFactoryName,true,
```

```
     ClassLoader.getSystemClassLoader());
    metricFactory = (ArmMetricFactory)
    metricFactoryClass.newInstance();
   }
  } catch ( Exception ex ) {
 ex.printStackTrace();
  }
}
```

Registering your application: ARM version 4

To inform ARM that an application should be monitored it needs to be registered.

Typically an Application registers itself each time the application is started while passing contextual information about itself that is used to identify that application. When the application shuts down it then tells the ARM consumer that the application is no longer running and any resources allocated on the local machine can be freed up for other uses.

ARM V4 is an objected oriented language based upon the Factories Pattern. An application definition is created using the Transaction Factory. An application definition is essentially the way you tell the ARM consumer about a new application it should monitor. Below is the signature of the method you use within the **ArmTransactionFactory** to create your application definition, ignore everything but the application name as we will talk about those optional parameters later.

Registering the application using Java

Create and return a new **ArmApplicationDefinition** object:

```
package org.opengroup.arm40.transaction;
```

Parameters:

- *name* The name of the **applicationDefinition**.
- *identityProperties* The identity properties associated with the **applicationDefinition**.
- *id* The id associated with the **applicationDefinition**.

Returns:

A new **ArmApplicationDefinition** object.

```
public ArmApplicationDefinition newArmApplicationDefinition(String name,
 ArmIdentityProperties identityProperties, ArmID id) ;

ArmTransactionFactory tranFactory = getTransactionFactory();

ArmApplicationDefinition myApplicationDefinition =
 tranFactory.newArmApplicationDefinition("WebSphere",null,null);
```

See the appendix for the details on how a transaction factory is obtained. This code creates an ARM application definition object for an application named *WebSphere*. This **ArmApplicationDefinition** object will be used later for a variety of method calls and is a good candidate for a singleton object.

Registering the application using C#

Create and return a new **ArmApplicationDefinition** object:

```
using OpenGroup.arm40.transaction;
```

Parameters:

- *name* The name of the application.
- *identityProperties* The identity properties associated with the application.
- *id* The id associated with the **applicationDefinition**.

Returns:

A new **ArmApplicationDefinition** object.

```
IArmApplicationDefinition newArmApplicationDefinition(string name,
IArmIdentityProperties identityProperties, IArmID id);

IArmTransactionFactory tranFactory = getTransactionFactory();

IArmApplicationDefinition myApplicationDefinition =
tranFactory.newArmApplicationDefinition("WebSphere",null,null);
```

See the appendix for the details on how a transaction factory is obtained. This code creates an ARM application definition object for an application named *WebSphere*. This **ArmApplicationDefinition** object will be used later for a variety of method calls and is a good candidate for a singleton object.

Registering the application using C/C++

Parameters:

- *appl_name* ASCIIZ pointer to the application name. Must not be NULL or an empty string.
- *input_app_id arm_id_t* structure pointer identifying a suggested UUID for this application, will be ignored by ITCAM
- *flags* Reserved parameter.
- *buffer4* Pointer to the subbuffers identifying the identity property names/values and context names etc. Ignore this for now; this will be discussed in a later section.
- *autput_app_id* The arm_id_t structure where the output UUID will be placed for use in future calls that require the application UUID.

Returns:

Non-zero on success.

```
arm_error_t arm_register_application(
 const arm_char_t *app_name,
 const arm_id_t *input_app_id,
 const arm_int32_t flags,
 const arm_buffer4_t *buffer4,
 arm_id_t *output_app_id);
```

The above API call returns a zero (0) on success, otherwise a return code on failure. The application name is passed in as the first parameter and is expected to be in the specified codeset (buffer4 element) or the default code set for that operating system. An application ID is returned in the output_app_id field and will be used in future API calls. This is a good candidate to be stored as a global variable or encapsulated inside a C++ singleton class.

Example:

```
arm_id_t appl_id;
arm_register_application("Websphere",NULL,0,NULL,&appl_id);
```

Essentially what you need to pass in to this call as a minimum is the application name and a variable for storing the generated application ID.

Creating an application instance: ARM version 4

Now that the ARM engine has been notified about the type of application it will be monitoring, you need to tell it what instance of that application the ARM calls will be made under.

An application instance is essentially an occurrence of an application and should be identified using repeatable information about the application that can be used to uniquely identify this instance from other instances. If there is no repeatable unique name for the server instance (like its cluster name) other than something like process ID, then it is best to not use an instance name to distinguish between instances. The arm engine always knows how to distinguish between instances of an application if they are running as different processes, because this information can be gathered during the API call by the ARM consumer. The application instance name is optional and is there for identifying logical differences between application instances or for the ability to distinguish between instances of a server in something like a cluster environment.

Creating an instance of an application is very straightforward and very similar to the way an application definition is created.

Creating an application instance using Java

```
/**
* Create and returns a new ArmApplication object.
*
* @param definition An ArmApplicationDefinition object.
* @param group The group name of this application.
* @param instance The instance name of this application.
* @param contextValues Array of context values.
* @return A new ArmApplication object.

*/
public ArmApplication newArmApplication(ArmApplicationDefinition
definition, String group, String instance, String[] contextValues);
```

This API is used by passing the application definition object first, which tells the ARM engine what application you are creating an instance of, and in addition allows you to optionally pass in an application group name and an application instance name.

Example:

```
ArmTransactionFactory tranFactory = getTransactionFactory();

ArmApplicationDefinition myApplicationDefinition =
tranFactory.newArmApplicationDefinition("WebSphere",null,null);

ArmApplication myApplicationInstance =
tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server",null);
```

Creating an application instance using C#

```
/**
* Create and returns a new ArmApplication object.
*
* @param definition An ArmApplicationDefinition object.
* @param group The group name of this application.
* @param instance The instance name of this application.
* @param contextValues Array of context values.
```

```
* @return A new ArmApplication object.
*/
IArmApplication newArmApplication(IArmApplicationDefinition definition,
string group, string instance,string[] contextValues);
```

This API is used by passing the application definition object first, which tells the ARM engine what application you are creating an instance of, and in addition allows you to optionally pass in an application group name and an application instance name.

Example:

```
IArmTransactionFactory tranFactory = getTransactionFactory();
IArmApplicationDefinition myApplicationDefinition =
tranFactory.newArmApplicationDefinition("WebSphere",null,null);
IArmApplication myApplicationInstance =
tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server"
,null);
```

Creating an application instance using C/C++

```
/** Creates an instance of an application.
* @param arm_id_t the application id of the application to start.
* @param arm_char_t the application group name in specified codeset.
* @param arm_char_t the application instance name in the specified
codeset.
* @param arm_int32_t flags, set to 0 for ITCAM.
* @param arm_buffer4_t* the arm 4 subbuffer information.
* @param arm_app_start_handle_t* pointer to the variable to store the
* generated application handle in.
*/
arm_start_application(
 const arm_id_t *app_id,
 const arm_char_t *app_group,
 const arm_char_t *app_instance,
 const arm_int32_t flags,
 const arm_buffer4_t *buffer4,
 arm_app_start_handle_t *app_handle);
```

Example:

```
arm_id_t appl_id;
arm_app_start_handle_t applicationStartHandle;
arm_register_application("Websphere",NULL,0,NULL,&appl_id );
arm_start_application(&appl_id ,"Cell/Node","Server",0,null,
 &applicationStartHandle);
```

Registering your transaction: ARM version 4

To tell ARM about the type of transaction that you will be executing, register the transaction. When you register a transaction you are telling the ARM engine the contextual information about a transaction that it can use to classify that transaction.

A transaction, in ARM terminology, is the section of work that should be monitored and timed uniquely by the ARM engine. Essentially what this means is that when a user hits a web page and gets the reply from the server this is single transaction from a user perspective, however from the server perspective this users view of the transaction is actually broken down into several or even 100's of units of work performed to resolve that request each of which in arm terminology is a transaction.

In ITCAM, a transaction from the user's perspective is the *edge transaction*. This is the first point in time where we begin monitoring the work that is being done to

service a user's request. Other units of work that are performed to process that edge transaction are called *subtransactions*.

The most important transaction from an ITCAM and user perspective is the edge transaction. The edge transaction is the overall transaction that tells you how the business process as a whole is performing. The edge transaction provides meaningful and useful contextual information that can be used by an ITCAM product user to create a management policy with which to manage that business process or type of request. In ITCAM for Transactions, only edge transactions can be shown in the Application Management Console workspaces, subtransactions are not displayed.

Transactions are usually registered one time at application startup, or when a class is loaded in Java using a static initializer for that class. Information that changes frequently from instance to instance is usually described at registration time and then has its actual values passed in at the time the transaction starts. However describing this information and the passing it in at start time is a topic that will be discussed in detail in the advanced section. In this section we will be assuming you will be performing transaction registration every time the transaction occurs.

Transactions should also be organized and formatted to be easily understood both by the ARM consumer as well as the user to views the data from the ARM consumer. How transaction names are constructed when taking the simplistic approach described here for ARM 4 and always for ARM 2 is extremely important and should involve a lot of thought.

Typically for subtransactions you name your transactions using:

Registering your transaction

Using Java:

```
packagename.classname.methodname(signature);
org.opengroup.arm40.ArmTransactionFactory.newArmApplicationDefinition(String,
ArmIdentityProperties, ArmID);
```

Using C#:

```
packagename.classname.methodname(signature);
IarmTransactionDefinition =
tranFactory.newArmTransactionDefinition(IArmApplicationDefinition
app,string name, IArmIdentityPropertiesTransaction identityProperties,
IArmID id);
```

Using C:

```
subdirectory/filename::functionname(signature);
src/Core/ARM/libarm4.cpp::arm_register_application(const arm_char_t*,
 const arm_id_t*,
 const arm_int32_t,
 const arm_buffer4_t*,
 arm_id_t *);
```

Using C++:

```
namespace::ClassName.MethodName(signature);
global::CArmAPI.arm_register_transaction(const arm_char_t*,
const arm_id_t*,
const arm_int32_t,
const arm_buffer4_t*,
arm_id_t *);
```

Edge transactions

For edge transactions you usually want to use more descriptive names from a business perspective so that your instrumentation does not look like just a full debug trace of your application which is not typically useful to someone who doesn't have the source code or inner implementation knowledge.

This might be something like:

URL

Script name and useful parameters

The Action being performed and the object it is being performed on

Internet Explorer for Microsoft Word

This type of naming convention is the most useful to the user, however it is also the naming convention that requires the most time, thought and effort. Typically for the most return on investment this type of naming convention should always be done on Edge Transactions, and only done on sub-transactions if you are using some kind of engine that decodes something and performs work based upon the values passed to it. A sub transaction named `runScript(String)` would not be useful to a user of your application, they would be more interested in `runScript(actualscriptname)`. In these examples, we registered a transaction for an object called `Foo` and method named bar with `signature (String)` or `Foo.bar(String)`.

Registering your transaction using Java

Registering a transaction is very similar to register an application. The main difference is it takes an application Definition object to which tells the ARM consumer what application definition this transaction should be associated with.

```
/**
* Create and return a new ArmTransactionDefinition object.
*
* @param app The ArmApplicationDefinition object that will be
assocated with this transaction definition.
* @param name The name of this transaction definition.
* @param identityProperties The IdentityProperties of this
transaction.
* @return A new ArmTransactionDefinition object.
*/
public ArmTransactionDefinition newArmTransactionDefinition(
 ArmApplicationDefinition app,
 String name,
 ArmIdentityPropertiesTransaction identityProperties,
 ArmID id);
```

For example:

```
ArmTransactionFactory tranFactory = getTransactionFactory();3
ArmApplicationDefinition myApplicationDefinition ;
ArmApplication myApplicationInstance;
ArmTransactionDefinition myTransactionDefinition;
myApplicationDefinition =
tranFactory.newArmApplicationDefinition("WebSphere",null,null);
myApplicationInstance
=tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server
",null);
myTransactionDefinition =
tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.bar(S
tring);",null,null);
```

Registering your transaction using C#

```
packagename.classname.methodname(signature);
IArmTransactionFactory tranFactory = getTransactionFactory();2
IArmApplicationDefinition myApplicationDefinition =
tranFactory.newArmApplicationDefinition("WebSphere",null,null);
IArmApplication myApplicationInstance =
tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server"
,null);
IArmTransactionDefinition tranDefinition =
tranFactory.newArmTransactionDefinition(
myApplicationDefinition,"Com.Foo.FooClass.FooMethod", null, null);
```

Registering your transaction using C/C++

```
* @param arm_id_t the application id returned from
arm_register_application
* @param arm_char_t the name of the transaction in specified character
set.
* @param arm_id_t* [optional] the input transaction ID.
* @param arm_int32_t flags, should be 0 for ITCAM.
* @param arm_buffer4_t [optional] arm 4 subbuffer.
* @param arm_id_t the output transaction ID.
**/
arm_error_t arm_register_transaction(
 const arm_id_t *app_id,
 const arm_char_t *tran_name,
 const arm_id_t *input_tran_id,
 const arm_int32_t flags,
 const arm_buffer4_t *buffer4,
 arm_id_t *output_tran_id);
```

For example:

```
arm_id_t appl_id;
arm_app_start_handle_t applicationStartHandle;
arm_id_t tran_id; arm_register_application("Websphere",NULL,0,NULL,&appl_id);
arm_start_application(&appl_id,"Cell/Node","Server",0,null,&applicationStartHandle);
arm_register_transaction(&appl_id,"Foo.bar(String);",NULL,0,NULL,&tran_id);
```

Starting and stopping a transaction: ARM version 4

After you have initialized the ARM engine with information about the application, you can start monitoring transactions.

The application now has a value in its singleton class, or has global variable information returned from the ARM consumer for each registration call. This information is used to start instances of transactions.

By registering, you have established a strong optimization. Information describing the application and its transactions is passed to the ARM engine, so information does not have to be collected and passed every time a transaction is started. This reduces the ARM instrumentation overhead during normal operation.

This section describes how to use the handles and objects returned by the ARM calls to execute transactions that will be monitored by the application.

When you stop a transaction you can specify whether the transaction succeeded, was aborted or whether it failed. When a transaction fails or is aborted, the ITCAM agents using ARM apply a threshold and track the transaction.

Note: While the use of correlators is not explored in this section, in reality you would never instrument an application that did not make use of correlation.

Without the use of correlators, the ARM consumer is unable to establish any relationship between transactions and its sub transactions.

To start and stop the transaction, call the `start()` and `stop()` method on the **`ArmTransaction`** object that is returned.

Using Java

Before you can start or stop an ARM 4 transaction there is another step that falls between registration of a transaction and starting or stopping a transaction. Since ARM 4 is object oriented it is necessary to create an object that will be used to contain information related to a started or stopped transaction. It's possible to create multiple of these if you wish so you can have multiple instances of the same transaction started at the same time. Typically however you will be creating a new instance of this object each time the transaction is started which is why I saved this step till this section. Also not that when creating this transaction object you specify what transaction definition it's associated with, and also specify what instance of the application it's associated with. This is done using the following method on the factory:

```
* Create and return a new ArmTransaction object.
*
* @param app The ArmApplication object.
* @param definition The ArmTransactionDefinition object.
* @return A new ArmTransaction object.
*/

public ArmTransaction newArmTransaction(ArmApplication app,
ArmTransactionDefinition definition);
```

For example:

```
ArmTransactionFactory tranFactory = getTransactionFactory();
ArmApplicationDefinition myApplicationDefinition ;
ArmApplication myApplicationInstance;
ArmTransactionDefinition myTransactionDefinition;
ArmTransaction myTransaction;

myApplicationDefinition =
 tranFactory.newArmApplicationDefinition("WebSphere",null,null);
myApplicationInstance =
 tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server",null);
myTransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.bar(String);
 ",null,null);
myTransaction =
 tranFactory.newArmTransaction(myApplicationInstance,myTransactionDefinition);
myTransaction.start();
myTransaction.stop(ArmConstants.STATUS_GOOD);
```

Using C#

```
* Create and return a new ArmTransaction object.
*
* @param app The ArmApplication object.
* @param definition The ArmTransactionDefinition object.
* @return A new ArmTransaction object.
*/
public ArmTransaction newArmTransaction(ArmApplication app,
ArmTransactionDefinition definition);
```

For example:

```
IArmTransactionFactory tranFactory = getTransactionFactory();5
IArmApplicationDefinition myApplicationDefinition ;
IArmApplication myApplicationInstance;
IArmTransactionDefinition myTransactionDefinition;
IArmTransaction myTransaction;

myApplicationDefinition =
 tranFactory.newArmApplicationDefinition("WebSphere",null,null);
myApplicationInstance=
 tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server",null);
myTransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.bar(String);
",null,null);
myTransaction =
 tranFactory.newArmTransaction(myApplicationInstance,myTransactionDefinition);
myTransaction.start();
myTransaction.stop(IArmConstants.STATUS_GOOD);
```

Using C, C++

```
/** starts a transaction.
* @param arm_app_start_handle_t the application instance handle.
* @param arm_id_t* the transaction id to start.
* @param arm_correlator_t* the parent correlator.
* @param arm_int32_t flags.
* @param arm_buffer4_t* the arm4 subbuffer.
* @param arm_tran_start_handle_t* the start handle that will be returned.
* @param arm_corrlator_t* the correlator that will be returned.
*/
ARM4_API_DYNAMIC(arm_error_t)arm_start_transaction(
const arm_app_start_handle_t app_handle,
const arm_id_t *tran_id,
const arm_correlator_t *parent_correlator,
const arm_int32_t flags,
const arm_buffer4_t *buffer4,
arm_tran_start_handle_t *tran_handle,
arm_correlator_t *current_correlator);

/** Stops a transaction
* @param arm_tran_start_handle_t the transaction instance to stop.
* @param arm_tran_status_t the status of the transaction.
* @param arm_int32_t flags
* @param arm_buffer4_t* the arm4 subbuffer.
*/

ARM4_API_DYNAMIC(arm_error_t)arm_stop_transaction(
const arm_tran_start_handle_t tran_handle,
const arm_tran_status_t tran_status,
const arm_int32_t flags,
const arm_buffer4_t *buffer4);
```

For example

```
arm_id_t appl_id;
arm_app_start_handle_t applicationStartHandle;
arm_id_t tran_id;
arm_tran_start_handle_t startHandle;
arm_register_application("Websphere",NULL,0,NULL,&appl_id);
arm_start_application(&appl_id,"Cell/Node","Server",0,null,&applicationStartHandle);
arm_register_transaction(&appl_id,"Foo.bar(String);",NULL,0,NULL,&tran_id);
arm_start_transaction(applicationStartHandle,&tran_id,NULL,0,NULL,startHandle,NULL);
arm_stop_transaction(startHandle,ARM_STATUS_GOOD,0,NULL);
```

Stopping your application: ARM version 4

When your application is shutting down, it is important to notify the ARM engine that the application will not be performing any more ARM calls.

Stopping your application for ARM 4 is a two stage process, one to stop the instance and another to deregister the application itself. The example in the following section illustrates this two-stage process, extending the previous example:

Using Java

```
ArmTransactionFactory tranFactory = getTransactionFactory();
ArmApplicationDefinition myApplicationDefinition ;
ArmApplication myApplicationInstance;
ArmTransactionDefinition myTransactionDefinition;
ArmTransaction myTransaction;
myApplicationDefinition =
 tranFactory.newArmApplicationDefinition("WebSphere",null,null);
myApplicationInstance =
 tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server",null);
myTransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.bar(String);
 ",null,null);
myTransaction =
 tranFactory.newArmTransaction(myApplicationInstance,myTransactionDefinition);
myTransaction.start();
myTransaction.stop(ArmConstants.STATUS_GOOD);
// stop your application instance.
myApplicationInstance.end();
// destroy your application definition.
myApplicationDefinition.destroy()
```

Using C#

```
IArmTransactionFactory tranFactory = getTransactionFactory();
IArmApplicationDefinition myApplicationDefinition ;
IArmApplication myApplicationInstance;
IArmTransactionDefinition myTransactionDefinition;
IArmTransaction myTransaction;
myApplicationDefinition =
tranFactory.newArmApplicationDefinition("WebSphere",null,null);
myApplicationInstance =
 tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server",null);
myTransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.bar(String);
",null,null);
myTransaction =
 tranFactory.newArmTransaction(myApplicationInstance,myTransactionDefinition);
myTransaction.start();
myTransaction.stop(ArmConstants.STATUS_GOOD);
// stop your application instance.
myApplicationInstance.end();
// destroy your application definition.
myApplicationDefinition.destroy()
```

Using C, C++

```
arm_id_t appl_id;
arm_app_start_handle_t applicationStartHandle;
arm_tran_start_handle_t startHandle;
arm_register_application("Websphere",NULL,0,NULL,&appl_id);
arm_start_application(&appl_id,"Cell/Node","Server",0,null,&applicationStartHandle);
arm_register_transaction(&appl_id,"Foo.bar(String);",NULL,0,NULL,&tran_id) ;
```

```
arm_start_transaction(applicationStartHandle,&tran_id,NULL,0,NULL,startHandle,NULL);
arm_stop_transaction(startHandle,ARM_STATUS_GOOD,0,NULL);
arm_stop_application(applicationStartHandle,0,NULL);
arm_destroy_application(&appl_id,0,NULL);
```

Passing correlators in arm_start_transaction calls

The examples in this section show a sequence of actions where Foo.Bar(String) calls method Foo.sub1(String), which in turn calls Foo.sub2(String) describing the following flow: **Foo.bar()** > **Foo.Sub1()** > **Foo.Sub2()**.

This scenario shows an overall transaction called: Foo.bar(String) which calls the Foo.sub1() method inside of that function. Then the Foo.sub1() method calls the Foo.sub2() method inside that function:

```
Foo.bar(String param) {
Foo.sub1();
}
Foo.sub1(String param) {
Foo.sub2();
}
Foo.sub2(String param)
{
```

Using Java

```
ArmTransactionFactory tranFactory = getTransactionFactory();
ArmApplicationDefinition myApplicationDefinition ;
ArmApplication myApplicationInstance;
ArmTransactionDefinition
 myTransactionDefinition,sub1TransactionDefinition,sub2TransactionDefinition;
ArmTransaction myTransaction,sub1Transaction,sub2Transaction;
myApplicationDefinition =
 tranFactory.newArmApplicationDefinition("WebSphere",null,null);
myApplicationInstance =
 tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server",null);
// Creation defintions of all your 3 transactions
myTransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.bar(String);
 ",null,null);
sub1TransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.sub1(String);
",null,null);
sub2TransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.sub2(String);
 ",null,null);
// Creation Instances of all 3 of your transactions
myTransaction =
 tranFactory.newArmTransaction(myApplicationInstance,myTransactionDefinition);
sub1Transaction =
 tranFactory.newArmTransaction(myApplicationInstance,sub1TransactionDefinition);
sub2Transaction =
 tranFactory.newArmTransaction(myApplicationInstance,sub2TransactionDefinition);
//Start your transactions
myTransaction.start();
sub1Transaction.start(myTransaction.getCorrelator());
sub2Transaction.start(sub1Transaction.getCorrelator());
//stop your transactions in the reverse order they started.
sub2Transaction.stop(ArmConstants.STATUS_GOOD);
sub1Transaction.stop(ArmConstants.STATUS_GOOD);
myTransaction.stop(ArmConstants.STATUS_GOOD);
```

Using C#

```
IArmTransactionFactory tranFactory = getTransactionFactory();
IArmApplicationDefinition myApplicationDefinition ;
IArmApplication myApplicationInstance;
```

```
IArmTransactionDefinition
 myTransactionDefinition,sub1TransactionDefinition,sub2TransactionDefinition;
IArmTransaction myTransaction,sub1Transaction,sub2Transaction;
myApplicationDefinition =
tranFactory.newArmApplicationDefinition("WebSphere",null,null);
myApplicationInstance
=tranFactory.newArmApplication(myApplicationDefinition,"Cell/Node","Server
",null);
// Creation defintions of all your 3 transactions
myTransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.bar(String);
 ",null,null);
sub1TransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.sub1(String);
 ",null,null);
sub2TransactionDefinition =
 tranFactory.newArmTransactionDefinition(myApplicationDefinition,"Foo.sub2(String);
 ",null,null);
// Creation Instances of all 3 of your transactions
myTransaction =
tranFactory.newArmTransaction(myApplicationInstance,myTransactionDefinition);
sub1Transaction =
 tranFactory.newArmTransaction(myApplicationInstance,sub1TransactionDefinition);
sub2Transaction =
 tranFactory.newArmTransaction(myApplicationInstance,sub2TransactionDefinition);
//Start your transactions
myTransaction.start();
sub1Transaction.start(myTransaction.getCorrelator());
sub2Transaction.start(sub1Transaction.getCorrelator());
//stop your transactions in the reverse order they started.
sub2Transaction.stop(IArmConstants.STATUS_GOOD);
sub1Transaction.stop(IArmConstants.STATUS_GOOD);
myTransaction.stop(IArmConstants.STATUS_GOOD);
```

Using C/C++

```
arm_id_t appl_id;
arm_app_start_handle_t applicationStartHandle;
arm_id_t tran_id,sub1_id,sub2_id;
arm_tran_start_handle_t startHandle,sub1Handle,sub2Handle;
arm_correlator_t rootCorrelator,sub1Correlator,sub2Correlator;
arm_register_application("Websphere",NULL,0,NULL,&appl_id);
arm_start_application(&appl_id,"Cell/Node","Server",0,null,&applicationStartHandle);
arm_register_transaction(&appl_id ,"Foo.bar(String);",NULL,0,NULL,&tran_id);
arm_register_transaction(&appl_id ,"Foo.sub1(String);",NULL,0,NULL,&sub1_id);
arm_register_transaction(&appl_id ,"Foo.sub2(String);",NULL,0,NULL,&sub2_id);
arm_start_transaction(applicationStartHandle,&tran_id ,NULL,0,NULL,&startHandle,
 &rootCorrelator );
arm_start_transaction(applicationStartHandle,&sub1_id ,&rootCorrelator,0,NULL,
 &sub1Handle,&sub1Correlator);
arm_start_transaction(applicationStartHandle,&sub2_id ,&sub1Correlator,0,NULL,
 &sub2Handle,&sub2Correlator );
arm_stop_transaction(sub2Handle,ARM_STATUS_GOOD,0,NULL);
arm_stop_transaction(sub1Handle,ARM_STATUS_GOOD,0,NULL);
arm_stop_transaction(startHandle,ARM_STATUS_GOOD,0,NULL);
```

Simplifying version 4 instrumentation

Simplify your ARM instrumentation by building a wrapper class and using the ARM 4 identity and context properties.

Using identity and context properties with an ARM wrapper

The examples in “Passing correlators in arm_start_transaction calls” on page 488 combined information about an application or transaction into a single string and passed it in as a single string. While this can be a simple way to pass in valuable context information about a transaction, it does have the following disadvantages:

- It's not easy to come up with a format for combining import contextual information into a single string.
- ARM consumers only have a combined “blob” of information with no real meaning attached to it other than it's all used to identify a transaction. This makes reporting on individual fields impossible or extremely challenging for Arm consumers because the combination of the fields is different for every single application and might even have multiple ways for every application.
- You have to register a transaction for every unique instrumentation point. Managing all of these objects can be tiresome and dirty up your applications code.

ARM 4 solves this problem by allowing the user to pass in all these different pieces of information in as properties. Properties are Name=Value pairs where the Name describes what the value means. Properties are limited to Strings, but it's easy to convert integers/etc to strings for passing in as properties.

You should only pass in property name/values that describe the type of transaction being performed. Identity and Context properties should not be used for passing metrics into the ARM api. Use of the ARM 4 metrics should be used for passing in metric information about a transaction.

The only real different between Identity Properties and Context Properties is when the value is passed into the Arm API call. Identity properties are used to pass in data to an ARM API when the value of that property is not going to change between occurrences of the transaction. Context values however are used for passing properties where the values change between occurrences of the transaction. There are a lot of different possibilities for using identity and context properties. Below I'll give an example of how we recommend doing your instrumentation. For details on how to pass properties into arm calls please see the opengroup ARM 4 API guide.

Example: Using Context properties to minimize the number of ArmTransactionDefinitions you need to create.

Let's say you want to implement monitoring of Servlets and the underlying support methods, however you don't want to be creating a ArmTransactionDefinition and then an ArmTransaction object for every servlet invocation. Additionally for every method you trace into you don't want to create a new ArmTransactionDefinition object in order to instrument that method. Likewise you want to minimize the code that must be placed within each instrumentation point. In this example we'll extend the ArmUtilities class to support some standard transaction types like Servlet entries and method trace transactions. Additionally we extend it to automatically manage your transaction objects and correlation. This should illustrate a very powerful and flexible way to

use context properties. Additionally we added threadlocal store for transactions so your instrumentation is a single start call upon method entry and a single stop call on exit.

Using the new ArmUtilities class you could instrument your servlet and supporting methods by using startServlet(..) and stopTransaction() at the entry and exit point of your servlet respectively. For the supporting methods for that servlet you just use the startMethod(...) and stopTransaction() methods at the entry and exit points of your instrumented methods. ARM instrumentation and correlation is fully supported in a multithreaded environment automatically.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.Properties;
import java.util.Stack;
import javax.servlet.http.HttpServletRequest;
import org.opengroup.arm40.metric.ArmMetricFactory;
import org.opengroup.arm40.tranreport.ArmTranReportFactory;
import org.opengroup.arm40.transaction.ArmApplication;
import org.opengroup.arm40.transaction.ArmApplicationDefinition;
import org.opengroup.arm40.transaction.ArmCorrelator;
import org.opengroup.arm40.transaction.ArmIdentityPropertiesTransaction;
import org.opengroup.arm40.transaction.ArmTransaction;
import org.opengroup.arm40.transaction.ArmTransactionDefinition;
import org.opengroup.arm40.transaction.ArmTransactionFactory;
/**
* Sample Arm Utility class that illustrates ways to encapsulate and simplify
* ARM instrumentation and correlation.
*
* @author Bret Patterson
*/
public class ArmUtilities {
private static ThreadLocal CorrelatorStack = new ThreadLocal() {
protected synchronized Object initialValue() {
return new Stack();
}
};
private static ThreadLocal transactionStack = new ThreadLocal() {
protected synchronized Object initialValue() {
return new Stack();
}
};
/** The ARM 4.0 Transaction Factory */
private static ArmTransactionFactory tranFactory;
/** The ARM 4.0 Transaction Report Factory */
private static ArmTranReportFactory reportFactory;
/** The ARM 4.0 Metric Factory */
private static ArmMetricFactory metricFactory;
/** The ARM 4.0 Application Definition for the TT product. */
private static ArmApplicationDefinition TheApplicationDefinition;
/** The ARM 4.0 Application Instance for the TT product. */
private static ArmApplication TheApplicationInstance;
private static ArmTransactionDefinition methodTransactionDefinition;
private static ArmTransactionDefinition servletTransactionDefinition;
private static final String[] SERVLET_CONTEXT_NAMES = { "RemoteAddress",
"RemotePort", "ServerAddress", "Port", "ReferrerURL", "Protocol",
"HostName", "Path", "File", "QueryString", "Anchor", "Method",
"Type", "Component" };
private static final int REMOTEADDRESS = 0;
private static final int PORT = 1;
private static final int PROTOCOL = 2;
private static final int HOSTNAME = 3;
private static final int PATH = 4;
private static final int FILE = 5;
private static final int QUERYSTRING = 6;
```

```
private static final int METHOD = 7;
private static final int TYPE = 8;
private static final int SERVLET_COMPONENT = 9;
private static final String[] METHOD_CONTEXT_NAMES = { "Component",
"Class", "Method" };
private static final String APPLICATION_NAME = null;
private static final String SERVLET_TRANSACTION_NAME = "Servlet";
private static final String METHOD_TRANSACTION_NAME = "Method";
private static final int TRANSACTION_COMPONENT = 0;
private static final int TRANSACTION_CLASS = 1;
private static final int TRANSACTION_METHOD = 2;
/**
* initializes the ARM factories and registers the supported transaction
* type definitions.
*/
static {
try {
Properties p = System.getProperties();
String keyTranFactoryClass = ArmTransactionFactory.propertyKey;
String tranFactoryName = p.getProperty(keyTranFactoryClass);
Class tranFactoryClass = Class.forName(tranFactoryName);
tranFactory = (ArmTransactionFactory) tranFactoryClass
.newInstance();
String keyTranReportFactoryClass = ArmTranReportFactory.propertyKey;
String reportFactoryName = p.getProperty(keyTranReportFactoryClass);
Class reportFactoryClass = Class.forName(reportFactoryName);
reportFactory = (ArmTranReportFactory) reportFactoryClass
.newInstance();
String keyMetricFactoryClass = ArmMetricFactory.propertyKey;
String metricFactoryName = p.getProperty(keyMetricFactoryClass);
Class metricFactoryClass = Class.forName(metricFactoryName);
metricFactory = (ArmMetricFactory) metricFactoryClass.newInstance();
registerApplication();
startApplication();
registerTransactions();
} catch (Exception ex) {
ex.printStackTrace();
}
}
/**
* This private method registers a generic servlet transaction and method
* transaction with context properties that can be used at transaction start
* time to identify transaction details.
*/
private static void registerTransactions() {
try {
ArmIdentityPropertiesTransaction identityServletProperties = tranFactory
.newArmIdentityPropertiesTransaction(null, null,
SERVLET_CONTEXT_NAMES, null);
ArmIdentityPropertiesTransaction identityMethodProperties = tranFactory
.newArmIdentityPropertiesTransaction(null, null,
METHOD_CONTEXT_NAMES, null);
if (TheApplicationDefinition != null) {
servletTransactionDefinition = tranFactory
newArmTransactionDefinition(TheApplicationDefinition,
SERVLET_TRANSACTION_NAME,
identityServletProperties, null);
methodTransactionDefinition = tranFactory
.
newArmTransactionDefinition(TheApplicationDefinition,
METHOD_TRANSACTION_NAME,
identityServletProperties, null);
}
} catch (Throwable ex) {
ex.printStackTrace();
}
}
```

```
/**
* Creates a simple Arm Application instance object for use by all
* transactions in this utility.
*/
private static void startApplication() {
if (tranFactory != null) {
TheApplicationInstance = tranFactory.newArmApplication(
TheApplicationDefinition, null, null, null);
} else {
TheApplicationDefinition = null;
TheApplicationInstance = null;
}
}
/**
* Creates a simple Arm Application definition object for use by all
* transactions in this utility
*/
private static void registerApplication() {
if (tranFactory != null) {
TheApplicationDefinition = tranFactory.newArmApplicationDefinition(
APPLICATION_NAME, null, null);
} else {
TheApplicationDefinition = null;
TheApplicationInstance = null;
}
}
/**
* This method is used to signal the start of a Servlet request. It handles
* all ARM related instrumentation for servlets. Effectively it does the
* following:
*
* Peeks at the top of the correlator stack and uses the correlator there as
* the parent correlator. Then it creates a transaction instance object and
* sets the context property values from the parameters passed in. Next it
* issues a transaction start call and pushes the output correlator onto the
* threadlocal transaction stack as well as the correlator on the
* threadlocal correlator stack.
*
* @param component
* The logical component name of this transaction. IE:
* SERVLET/EJB/JDBC/etc\
* @param request
* the J2EE servlet request object to use to pass in transaction
* information as context properties.
*/
public static void startServlet(String component, HttpServletRequest req) {
if (tranFactory != null) {
try {
ArmTransaction transaction = tranFactory.newArmTransaction(
TheApplicationInstance,
servletTransactionDefinition);
transaction.setContextValue(REMOTEADDRESS, req.getRemoteAddr());
transaction.setContextValue(PORT, Integer.toString(req
.getServerPort()));
transaction.setContextValue(PROTOCOL, req.getProtocol());
transaction.setContextValue(HOSTNAME, req.getServerName());
transaction.setContextValue(PATH, req.getPathTranslated());
transaction.setContextValue(FILE, req.getServletPath());
transaction.setContextValue(QUERYSTRING, req.getQueryString());
transaction.setContextValue(METHOD, req.getMethod());
transaction.setContextValue(TYPE, req.getScheme());
transaction.setContextValue(SERVLET_COMPONENT, component);
String user = req.getRemoteUser();
transaction.setUser(tranFactory.newArmUser(user == null ? "*"
: user, null));
ArmCorrelator parentCorrelator = getCorrelator();
if (parentCorrelator == null) {
```

```
transaction.start();
} else {
transaction.start(parentCorrelator);
}
pushTransaction(transaction);
pushCorrelator(transaction.getCorrelator());
} catch (Throwable ex) {
ex.printStackTrace();
}
}
}
/**
* This method is used to signal the start of a method. It handles all ARM *
* related instrumentation for methods. Peeks at the top of the correlator
* stack and uses the correlator there as the parent correlator. Then it
* creates a transaction instance object and sets the context property
* values from the parameters passed in. Next it issues a transaction start
* call and pushes the output correlator onto the threadlocal transaction
* stack as well as the correlator on the threadlocal correlator stack.
*
* @param component
* The logical component name of this transaction. IE:
* SERVLET/EJB/JDBC/etc
* @param classname
* The classname containing the method being executed.
* @param methodsignature
* the full method signature of the method being executed.
*/
public static void startMethod(String component, String classname,
String methodSignature) {
if (tranFactory != null) {
try {
ArmTransaction transaction = tranFactory.newArmTransaction(
TheApplicationInstance,
methodTransactionDefinition);
transaction.setContextValue(TRANSACTION_COMPONENT,
component);
transaction.setContextValue(TRANSACTION_CLASS, classname);
transaction
.setContextValue(TRANSACTION_METHOD,
methodSignature);
ArmCorrelator parentCorrelator = getCorrelator();
if (parentCorrelator == null) {
transaction.start();
} else {
transaction.start(parentCorrelator);
}
pushTransaction(transaction);
pushCorrelator(transaction.getCorrelator());
} catch (Throwable ex) {
ex.printStackTrace();
}
}
}
/**
* This method stops the transaction that is currently on the top of the
* stack. It can be used to simplify arm instrumentation for single threaded
* applications. This method pops the transaction off of the top of the
* stack and then stops that transaction. It then pops the top correlator
* off of the stack.
*
* @param The
* status code for the transaction take from:
* org.opengroup.arm40.transaction.ArmConstants
*/
public static void stopTransaction(int status) {
if (tranFactory != null) {
```

```
try {
ArmTransaction transaction = popTransaction();
transaction.stop(status);
popCorrelator();
} catch (Exception ex) {
ex.printStackTrace();
}
}
}
/**
* Get the correlator from the top of the correlator stack.
*
* @return the correlator at the top of the correlator stack.
*/
public static ArmCorrelator getCorrelator() {
Stack corStack = (Stack) CorrelatorStack.get();
ArmCorrelator parentCorrelator;
if (corStack.size() > 0) {
// get a copy of the parent correlator.
parentCorrelator = (ArmCorrelator) corStack.peek();
} else {
parentCorrelator = null;
}
return parentCorrelator;
}
/**
* Push a correlator onto the top of the correlator stack.
*
* @param the
* correlator to place on the top of the correlator stack.
*/
public static void pushCorrelator(ArmCorrelator currentCorrelator) {
// get this threads stack
Stack corStack = (Stack) CorrelatorStack.get();
// push the current correlator onto the top of the stack
corStack.push(currentCorrelator);
// update the thread local variable
CorrelatorStack.set(corStack);
}
/**
* Pops a correlator off the top of the correlator stack.
*
*/
public static void popCorrelator() {
// get this threads stack
Stack corStack = (Stack) CorrelatorStack.get();
if (corStack.size() > 0) {
// pop the current correlator off the top of the stack
corStack.pop();
// update the thread local variable
CorrelatorStack.set(corStack);
}
}
/**
*
* @return the ArmTransaction on the top of the transaction stack.
*
*/
private static ArmTransaction getTransaction() {
Stack tranStack = (Stack) transactionStack.get();
ArmTransaction parentTransaction;
// get a copy of the parent transaction
if (tranStack.size() > 0) {
parentTransaction = (ArmTransaction) tranStack.peek();
} else {
parentTransaction = null;
}
```

```
return parentTransaction;
}
/**
* Pops a transaction off of the top of the transaction stack and returns
* it.
*
* @return the ArmTransaction object that was popped off of the top of the
* stack.
*
*/
private static ArmTransaction popTransaction() {
ArmTransaction transaction = null;
// get this threads stack
Stack tranStack = (Stack) transactionStack.get();
//pop the current transaction off the top of the stack
if (tranStack.size() > 0) {
transaction = (ArmTransaction) tranStack.pop();
// update the thread local variable
transactionStack.set(tranStack);
} else {
transaction = null;
}
return transaction;
}
/**
* Pushes an ArmTransaction object onto the transaction stack.
*
* @param currentTransaction
* the ArmTransaction object to push onto the stack.
*/
private static void pushTransaction(ArmTransaction currentTransaction) {
// get this threads stack
Stack tranStack = (Stack) transactionStack.get();
// push the current correlator onto the top of the stack
tranStack.push(currentTransaction);
// update the thread local variable
transactionStack.set(tranStack);
}
} // end of ArmUtilities class
```

Performance considerations

Adding instrumentation to an application incurs a performance overhead. Control the instrumentation and diagnostic detail level to control the performance overhead incurred by monitoring the application.

Instrumentation detail control

One method of controlling the overhead is to make the number of points monitored in your application configurable. Using instrumentation detail control, you provide a configuration value that enables the application user to tune the instrumentation to different levels. For example:

- `Off` - No instrumentation occurs.
- `Entry/Exit into the application only` - Monitor only entry and exit points for the application, not entry and exit points for a function. That is, monitor only when a thread starts or when an API call is made to another application.
- `Medium Trace Level` - Only component level entry and exit points are traced.
- `High Trace Level` - All non-trivial function level entry and exit points inside functions are traced. Have your application instrument all non-trivial function entry and exit points. Then, check the configuration value of each function and

abort the instrumentation points of those that do not meet the specified level, before gathering information to make the transaction start call.

Diagnostic information control

Another method of controlling the overhead is to make the amount of metric information passed for each ARM call configurable. Gathering diagnostic information and passing all metric information to the ARM agent can be an expensive process due to the volume of data and frequency of its passing. Enabling the user to specify how much diagnostic information is gathered enables them to balance the ease of debugging a problem with the base instrumentation overhead. For example:

- `Off` - No diagnostic information is gathered.
- `Low Diagnostic information` - Only a small amount of information is gathered.
- `Medium Trace Level` - Only information that is of reasonable size and reasonably fast to gather is done.
- `High Trace Level` - All information relevant to this transaction is gathered.

Alternate language bindings

The ARM API includes bindings for the C, C++, and Java languages, but can be called from any language. If you are using a language other than C, C++, or Java, you should call the ARM API as an external C function from your programming language.

Making ARM API calls from PowerBuilder applications

This section uses a PowerBuilder application as an example of how to call the ARM API using a language other than C or C++.

PowerBuilder provides its own development language called PowerScript. You can make all of the ARM API calls in PowerScript following its syntax rules. Here is an example of how the ARM API calls can be made using a PowerBuilder script. After each call this script pops up a window to display the handle. This can be useful during initial development and testing.

```
arm_application = arm_init("new_account", "*",0,0,0)
messagebox("arm_init", string(arm_application))

arm_transaction = arm_getid(arm_application, "open", "", 0,0,0)
messagebox("arm_getid", string(arm_transaction))

arm_start_val = arm_start(arm_transaction, 0,0,0)
t = getapplication()
messagebox("arm_start", string(arm_start_val))

t.af_opencustomer(s)

arm_stop = arm_stop(arm_start_val, 0,0,0,0)
messagebox("arm_stop", string(arm_stop))

arm_end_val = arm_end(arm_transaction,0,0,0)
messagebox("arm_transaction", string(arm_end_val))
```

For the ARM function calls to be resolved, you must declare the ARM API functions to PowerBuilder as Global External Functions.

In a PowerBuilder window click Declare on the toolbar and open the Global External Function editor. Fill in the ARM function prototypes as shown in the following example.

Note: These declarations are case sensitive. Everything must be in lowercase.

```
function long arm_init(string name,string user_id,long flag,long data,long
size)
library "libarm32.dll"

function long arm_getid(long id,string name,string detail,long flag,long
data, long size) library "libarm32.dll"

function long arm_start(long id,long flag,long data,long size) library
"libarm32.dll"

function long arm_update(long handle,long flag,long data,long size)
library "libarm32.dll"

function long arm_stop(long handle,long status,long flag,long data,long
size)
library "libarm32.dll"

function long arm_end(long id,long flag,long data,long size) library
"libarm32.dll"
```

When the Powerbuilder application runs, the ARM calls that it makes are resolved in the external function definitions and result in ARM API calls being made to the ARM client agent.

Making ARM API calls from Visual Basic applications

The following steps are required to use ARM API calls from a Visual Basic application:

1. Declare the ARM API functions to Visual Basic as DLL procedures by loading the file form1.frm from the ARM Version 2 Software Developer's Kit (SDK) into your Visual Basic development environment.
2. Call the DLL procedures in your Visual Basic program.

Using the ARM Version 2 Software Developer's Kit

The ARM Software Developer's Kit (SDK) enables you to build and run application programs that are instrumented for ARM, even if you do not yet have an ARM agent.

You can download a free copy of the SDK from http://www.opengroup.org/arm/.

The SDK includes the header and library files necessary to compile an ARMinstrumented application that is written in C or C++, and a no-operation (NOP) runtime library.

Note: The no-operation (NOP) runtime library is available only in the ARM Version 2 SDK.

IBM standard for passing correlators

Using JMS

To pass a correlator to a J2EE-instrumented Application server from your ARM instrumented application, add the **ARM_CORRELATOR** parameter and the hex string for the correlator to the message being sent. Where:

- **Property Name**: must be set to ARM_CORRELATOR
- **Value**: must be set to the hex string representation of the correlator

If the application server receives a message with an **ARM_CORRELATOR** parameter, the embedded correlator is used to begin monitoring the transaction within the J2EE server.

For example:

```
message.setStringProperty(
"ARM_CORRELATOR",ToHexString(jmsCorrelator.getBytes()) );
```

Note: See the JMSCorrelatorPassing.java file in the JMSExample directory to see complete source code. These files are located at the following website (this example uses ITCAM ARM 2 interfaces): Correlator Passing

Using HTTP

To pass a correlator to a J2EE-instrumented Application server from your ARM instrumented application, modify the HTTP request header to add an element in *Key:Value* format. Where:

- **Key**: must be set to ARM_CORRELATOR
- **Value**: must be set to the hex string representation of the correlator

The application server that receives the HTTP requests looks for the **ARM_CORRELATOR** key in the HTTP header elements and begins monitoring the transaction in the J2EE server.

For example:

```
Socket socket = new Socket(host,port_number);
OutputStream os = socket.getOutputStream();
boolean autoflush = true;
PrintWriter out = new PrintWriter(os, autoflush);
out.println("GET " + uri + " HTTP/1.1");
out.println("Host: " + host);
out.println("Connection: Close");
out.println("ARM_CORRELATOR: " + ToHexString(httpCorrelator.getBytes()));
```

Using SOAP

To pass a correlator from your ARM-instrumented application to a J2EE-instrumented WebSphere Application server, modify the SOAP header to add a header element that contains the correlator. To do this, a SOAP Handler needs to be implemented.

Note: J2EE-instrumented Weblogic Application servers do not support this functionality.

The application server that receives the SOAP message looks for the required element in the SOAP header, and if it is found, the embedded correlator is used to activate monitoring the transaction within the J2EE server.

For example:

```
protected static final String REQMETRICS_NS_URI =
"http://websphere.ibm.com";
protected static final String REQMETRICS_PREFIX = "reqmetrics";
protected static final String REQMETRICS_ELEMENT = "correlator";
protected final static String REQMETRICS_ACTOR_URI = "reqmetricsURI";
/**
* Add correlator to outgoing message.
*/
public boolean handleRequest(MessageContext mc) {
soapTxn.start();
soapCorrelator = soapTxn.getCorrelator();
correlatorHexString =
ToHexString(soapCorrelator.getBytes());
addCorrelator((SOAPMessageContext)mc);
return true;
}
/**
* Add correlator to the SOAP message header
*/
public void addCorrelator(SOAPMessageContext mc) {
try {
SOAPMessage msg = mc.getMessage();
SOAPPart part = msg.getSOAPPart();
SOAPEnvelope envelope = part.getEnvelope();
SOAPHeader header = envelope.getHeader();
if (header == null) {
header = envelope.addHeader();
}
// create new header element, add correlator to it
Name temp_name = envelope.createName(REQMETRICS_ELEMENT,
REQMETRICS_PREFIX,
REQMETRICS_NS_URI);
SOAPHeaderElement she = header.addHeaderElement(temp_name);
// set element value is the correlator
she.addTextNode(correlatorHexString);
she.setActor(REQMETRICS_ACTOR_URI);
} catch (Exception e) {
e.printStackTrace();
}
}
```

See the `SOAPClientHandler.java` file in the `SOAPExample/src` directory for a complete example of a SOAP handler. These files are located at the following website (the examples are based on ARM 2 interfaces): Correlator Passing

Using RMI for WebSphere Application Server Version 5 and later

To pass a correlator from an ARM-instrumented application to a J2EE-instrumented WebSphere Application Server (WAS) Version 5 and later, add it to the RMI request as **`ServiceContext`** data. The **`ServiceContext`** key used to send the correlator is an ID assigned to Response Time Tracking by WebSphere (0x49424d1e), and the value is a byte array which represents the correlator. An RMI Interceptor must be implemented to modify RMI requests to add to the correlator. The interceptor has to implement the `org.omg.PortableInterceptor.ClientRequestInterceptor` interface.

The WebSphere Application Server Version 5 server that receives the RMI-IIOP requests looks for the ID (0x49424d1e) in the ServiceContext of the request, and if it is found, the embedded correlator is used to continue monitoring the transaction.

For example:

```
public static final int TMTP_ID = 0x49424d1e;
public void send_request(ClientRequestInfo cri) throws ForwardRequest
{
try {
 rmiTxn.start();
 rmiCorrelator = rmiTxn.getCorrelator();
 byte[] corrBytes = rmiCorrelator.getBytes();
 //This is an IBM specific class
 ExtendedClientRequestInfo ri = (ExtendedClientRequestInfo) cri;
 byte data[] = null;
 ORB theORB = (ORB) ((LocalObject) ri)._orb();
 CDROutputStream cos = ORB.createCDROutputStream(theORB);
 cos.putEndian();
 int length = corrBytes.length;
 cos.write_string("" + length);
 cos.write_octet_array(corrBytes, 0, corrBytes.length);
 data = cos.toByteArray();
 // Add the ServiceContext with TMTP_ID and Correlator to the
 // request to be picked up by the Server side
 ServiceContext sc = new ServiceContext(WASRMIInterceptor.TMTP_ID,data);
 ri.add_request_service_context(sc, true);
  } catch (Exception e) {
  e.printStackTrace();
  }
 }
```

See the `WASRMIInterceptor.java` file in the `RMIExample` directory to see a complete example of a RMI Interceptor for sending correlators to WebSphere Application Server Version 5 and above. These files are located at the following website: Correlator Passing

Using RMI for Weblogic

To pass a correlator to a J2EE-instrumented WebLogic Application Server use the **`weblogic.trace.Trace`** class. The approach below works only if it is run in a Weblogic container.

For example:

```
rmiTxn.start();
rmiCorrelator = rmiTxn.getCorrelator();
byte[] corrBytes = rmiCorrelator.getBytes();
T3Payload payload = new T3Payload(corrBytes);
weblogic.trace.Trace.beginTrace(T3Payload.serialize(payload));
// T3Payload class
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
class T3Payload implements Serializable {
 private static int MONITORING_APP_ID = 1;
 private String hostName = null;
 private int appId = -1;
 private String threadName = null;
 private byte[] correlator = null;
 T3Payload(byte[] c)
 {
```

```
this("localhost",MONITORING_APP_ID,Thread.currentThread().getName(),c);
}
T3Payload(String h, int id, String t, byte[] c)
{
 hostName = h;
 appId = id;
 threadName = t;
 correlator = c;
}
static byte[] serialize(T3Payload p)
{
 byte[] rv = null;
 if (p != null) {
  try {
   // Create a Output stream to serialize to
   ByteArrayOutputStream bos = new ByteArrayOutputStream();
   ObjectOutputStream s = new ObjectOutputStream(bos);
   // Serialize the payload
   s.writeObject(p);
   rv = bos.toByteArray();
   // Reset and flush the ObjectOutputStream
   s.reset();
   s.flush();
   bos = null;
   s = null;
  } catch (IOException e) {
   e.printStackTrace();
   rv = null;
  }
 }
 return rv;
}
static T3Payload deserialize(byte[] c)
{
 T3Payload rv = null;
 if (c != null) {
  try {
   //Create a Input stream to deserialize from
   ByteArrayInputStream bis = new ByteArrayInputStream(c);
   ObjectInputStream s = new ObjectInputStream(bis);
   // De-serialize the payload
   rv = (T3Payload) s.readObject();
   //To satisfy a jtest rule.
   if (s.markSupported()) {
    s.reset();
   }
   bis = null;
   s = null;
  } catch (Exception e) {
   e.printStackTrace();
   rv = null;
  }
 }
 return rv;
}
}
```

libSWARM function calls

Rational Robot scripts are automatically SWARM instrumented when the scripts are uploaded to the management server. This section provides a reference for the libSWARM library.

arm_set_app_name_rt

- GUI:

```
Declare Sub arm_set_app_name_rt Lib "libSWARM32"
(ByVal applicationName As String)
```

- VU:

```
external_C proc arm_set_app_name_rt(applicationName)
string applicationName;
{}
```

Sets the application name for all subsequent SWARM calls.

The *applicationName* variable specifies the value for the application name. This represents the application under test and not the name of the Robot recording. Where:

- *applicationName* is the name of the application under test

Note: If this Robot recording is to be used with Generic Windows monitoring, there is no need to make this call in your script as the default application name is automatically set to `GenWin`.

Returns zero (0) if successful and a negative value if an error occurs.

For example, if the application under test is `Online Banking`:

- GUI:

```
arm_set_app_name_rt "Online Banking"
```

- VU:

```
arm_set_app_name_rt("Online Banking");
```

arm_set_name_rt

- GUI:

```
Declare Sub arm_set_name_rt Lib "libSWARM32" (ByVal
defaultTransactionName As String)
```

- VU:

```
external_C proc arm_set_name_rt(defaultTransactionName);
string defaultTransactionName;
{}
```

Sets the base ARM pattern for all subsequent `libSWARM arm_start_rt()` calls.

The *defaultTransactionName* variable specifies the value for the default transaction name. A naming pattern must be adopted for the libSWARM calls for ARM to recognize a Robot transaction. The pattern requires the ARM calls to begin with the script name followed by any characters: `ScriptName.*`. Where:

- *ScriptName* is the name of the recorded GUI or VU test script
- .* is zero or more characters

Returns zero (0) if successful and a negative value if an error occurs.

For example, if a recorded test is named AccountLoginTest:

- GUI:

```
arm_set_name_rt("AccountLoginTest")
```

- VU:

```
arm_set_name_rt("AccountLoginTest");
```

arm_start_rt

- GUI:

```
Declare Function arm_start_rt Lib "libSWARM32" () As Integer
```

- VU:

```
external_C int func arm_start_rt()
{}
```

Note: The arm_start_rt method exists to simplify test instrumentation. To provide a descriptive name to a transaction step, use arm_start_with_name_rt.

Signals the start of an ARM transaction, and assigns it a generated name based on the current *defaultTransactionName* value. The generated name is formatted as follows: TransactionName__LevelX_StepY. Where:

- *TransactionName* is the name of the recorded test
- *LevelX* is the ARM nesting level
- *StepY* is the step number within the nesting level

AccountLoginTest__Level2 _Step4 for example, is the fourth step on the second level of the recorded AccountLoginTest test.

Returns zero (0) if successful and a negative value if an error occurs.

For example:

- GUI:

```
arm_start_rt
```

- VU:

```
arm_start_rt();
```

arm_start_with_name_rt

- GUI:

```
Declare Function arm_start_with_name_rt Lib
"libSWARM32" (ByVal transactionName As String, ByVal
transactionGroupName As String) As Integer)
```

- VU:

```
external_C int func
arm_start_with_name_rt(transactionName,
transactionGroupName)
string transactionName;
string transactionGroupName;
{}
```

Signals the start of an ARM transaction and assigns it the specified name and group name, where:

- *transactionName* is the name to assign to the transaction. For the top level ARM transaction the *transactionName* must match the test name like arm_set_name_rt. For sub transactions the *transactionName* does not need to match the test name.

- `groupName` is the name to assign to the transaction group. This is used for grouping similar transactions for reporting purposes.

Returns zero (0) if successful and a negative value if an error occurs.

For example, if the transaction is named `AccountLoginTest` and belongs to the Login group:

- GUI:

```
arm_start_with_name_rt "AccountLoginTest", "Login"
```

- VU:

```
arm_start_with_name_rt("AccountLoginTest", "Login");
```

arm_start_with_parent_rt

- GUI:

```
Declare Function arm_start_with_parent_rt Lib
"libSWARM32" (ByVal parent_transaction_name As String,
ByVal transaction_name As String, ByVal group_name As
String) As Integer
```

- VU:

```
external_C int func
arm_start_with_parent_rt(parentTransactionName,
transactionName, group)
string parentTransactionName;
string transactionName;
string group;
{}
```

Signals the start of an ARM transaction and assigns it the specified name and group name. This function can be used to specifically identify the parent transaction in the case of asynchronous transactions or transactions that are called out of order. Where:

- *parentTransactionName* is the name of the parent transaction for correlation purposes. This must be an existing transaction that is already started but not yet stopped.
- *transactionName* is the name to assign to the transaction. For the top level ARM transaction the *transactionName* must match the test name like `arm_set_name_rt`. For sub transactions the *transactionName* does not need to match the test name.
- `groupName` is the name to assign to the transaction group. This is used for grouping similar transactions for reporting purposes.

Returns zero (0) if successful and a negative value if an error occurs.

For example, if there is a parent transaction named `CheckAccount` with a child transaction named `AccountLoginTest` that belongs to the Login group:

- GUI:

```
arm_start_with_parent_rt "CheckAccount", "AccountLoginTest",
"Login"
```

- VU:

```
arm_start_with_parent_rt("CheckAccount", "AccountLoginTest",
"Login");
```

arm_stop_rt

- GUI:

```
Declare Function arm_stop_rt Lib "libSWARM32" () As
Integer
```

- VU:

```
external_C int func arm_stop_rt()
{}
```

Signals the end of an ARM transaction. This does nothing when called prior to an associated START call. The status of the stopped transaction is set to ARM_GOOD. See the arm_stop_with_status_rt command for more information.

Returns zero (0) if successful and a negative value if an error occurs.

For example:

- GUI:

```
arm_stop_rt
```

- VU:

```
arm_stop_rt();
```

arm_stop_failed_rt

- GUI:

```
Declare Function arm_stop_failed_rt Lib "libSWARM32"
() As Integer
```

- VU:

```
external_C int func arm_stop_failed_rt()
{}
```

Signals the end of an ARM transaction. This does nothing when called prior to an associated START call. The status of the stopped transaction is set to **ARM_FAILED**.

Returns zero (0) if successful and a negative value if an error occurs.

For example:

- GUI:

```
arm_stop_failed_rt
```

- VU:

```
arm_stop_failed_rt();
```

arm_stop_with_status_rt

- GUI:

```
Declare Function arm_stop_with_status_rt Lib
"libSWARM32" (ByVal status As Integer) As Integer
```

- VU:

```
external_C int func arm_stop_with_status_rt(status)
int status;
{}
```

Signals the end of an ARM transaction. This does nothing when called prior to an associated START call. The status of the stopped transaction is set to the specified status value. Where:

- *status* is the status to use for the transaction stop call

Returns zero (0) if successful and a negative value if an error occurs.

For example:

- GUI:

```
arm_stop_with_status_rt ARM_GOOD
```

- VU:

```
arm_stop_with_status_rt (ARM_GOOD);
```

ARM transaction status codes used by **arm_stop_with_status_rt**:

- ARM_GOOD = 0 indicates a successful transaction
- ARM_ABORT = 1 indicates an aborted transaction
- ARM_FAILED = 2 indicates a failed transaction
- ARM_IGNORE = 3 indicates an ignored transaction

arm_stop_all_with_status_rt

- GUI:

```
Declare Function arm_stop_all_with_status_rt Lib
"libSWARM32" (ByVal status As Integer) As Integer
```

- VU:

```
external_C int func
arm_stop_all_with_status_rt(status)
int status;
{}
```

Signals the stop of all currently started ARM transactions. This function can be used to cleanup and close all remaining started transactions prior to exiting the script. Where:

- *status* is the status to use for all transaction stop calls

Returns zero (0) if successful and a negative value if an error occurs.

For example:

- GUI:

```
arm_stop_all_with_status_rt ARM_GOOD
```

- VU:

```
arm_stop_all_with_status_rt(ARM_GOOD);
```

ARM transaction status codes used by **arm_stop_all_with_status_rt**:

- ARM_GOOD = 0 indicates a successful transaction
- ARM_ABORT = 1 indicates an aborted transaction
- ARM_FAILED = 2 indicates a failed transaction
- ARM_IGNORE = 3 indicates an ignored transaction

arm_stop_with_name_rt

- GUI:

```
Declare Function arm_stop_with_name_rt Lib
"libSWARM32" (ByVal transaction_name As String, ByVal
status As Integer) As Integer
```

- VU:

```
external_C int func
arm_stop_with_name_rt(transactionName, status)
string transactionName;
int status;
{}
```

Signals the stop of an ARM transaction by name. This function can be used to specifically identify the transaction to stop in the case of asynchronous transactions or transactions that are called out of order. Where:

- *transactionName* is the name of the transaction to stop
- *status* is the status to use for all transaction stop calls

Returns zero (0) if successful and a negative value if an error occurs.

For example, if the transaction is named *AccountLoginTest*:

- GUI:

```
arm_stop_with_name_rt "AccountLoginTest", ARM_GOOD
```

- VU:

```
arm_stop_with_name_rt("AccountLoginTest", ARM_GOOD);
```

ARM transaction status codes used by **arm_stop_with_name_rt**:

- `ARM_GOOD = 0` indicates a successful transaction
- `ARM_ABORT = 1` indicates an aborted transaction
- `ARM_FAILED = 2` indicates a failed transaction
- `ARM_IGNORE = 3` indicates an ignored transaction

arm_copy_correlator_hex_rt

- GUI: HTTP correlator passing is not supported for Rational Robot GUI.
- VU:

```
external_C proc
arm_copy_correlator_hex_rt(outCorrelatorHexString)
reference string outCorrelatorHexString;
{}
```

Accesses the current ARM Correlator. This returns the correlator as a string that contains the hex value of the ARM correlator. This string can be added to an HTTP request to correlate with components down stream from the request (QoS, J2EE, DB2, CICS, and so on). The header format is **ARM_CORRELATOR**, where:

- *Correlator* is the hex value returned
- *outCorrelatorHexString* is the string variable (defined in the accessing VU Script) to store the hex string denoting the current ARM correlator

For example:

- VU:

```
string arm_corr;
arm_copy_correlator_hex_rt(&arm_corr);
sprintf(&http_corr, "ARM_CORRELATOR: %s", arm_corr);
http_request ["VuIEIbm~010"]
 "GET /data/js/survey/esites/popup.js HTTP/1.1\r\n"
 "Accept: */*\r\n"
 "Referer: " + SgenURI_003 + "\r\n"
 /*  "Referer: http://www.ibm.com/us/" */
 "Accept-Language: en-us\r\n"
 "Accept-Encoding: gzip, deflate\r\n"
 "If-Modified-Since: Wed, 09 Feb 2005 20:12:31 GMT\r\n"
 "If-None-Match: \"260d9-1649-420a6eaf\"\r\n"
 "User-Agent: Mozilla/4.0 (compatible; MSIE 6.0/6.1; Windows NT 5.1; .NET CLR"
 " 1.1.4322)\r\n"
 "Host: www.ibm.com\r\n"
```

```
 "Connection: Keep-Alive\r\n"
 "Cookie: w3ibmProfile=20011110162543016112757339l7| gAME|897|BNT|null\r\n"
/*TMTP: Added ARM Correlator to HTTP Header*/
 + http_corr + "\r\n" "\r\n";
```

arm_get_status_rt

- GUI:

```
Declare Function arm_get_status_rt Lib "libSWARM32" ()
As Integer
```

- VU:

```
external_C int func arm_get_status_rt()
{}
```

Returns the status code of the last SWARM function call.

For example:

- GUI:

```
DIM status AS Integer
...
status = arm_get_status_rt
```

- VU:

```
int status;
...
status = arm_get_status_rt();
```

Appendix G. ARM response codes

The ARM response codes have been customized for ITCAM for Transactions.

General response codes

Table 83 describes the general ARM response codes.

Table 83. General ARM response codes

Code	Description
0	The ARM call was successful.
-1	Invalid input parameter.
-2	The ARM agent is not running.
-3	Unable to communicate with the ARM agent.
-4	Error found inside the ARM agent. An error message has been logged.
-5	The API calls you are using are not supported by this agent.
-6	The heap is out of memory.
-7	The library initialization failed, so it is disabled.
-8	The IPC Queue is full.
-9	An IPC call failed.
-249	An unspecific error occurred while processing an ARM call.
-1000	Invalid parameter passed to ARM.
-1001	Transaction, Application, or Metric name passed to ARM was an empty string.
-1002	Transaction, Application, or Metric name passed to ARM contains too many characters.
-1003	Error performing character set conversion.
-1100	Application ID passed to ARM is not valid.
-1101	Output application ID passed to arm is not valid. (may be null)
-1102	Output application handle passed to ARM is not valid. (may be null)
-1200	Transaction ID passed to ARM is not valid.
-1201	Transaction ID passed to ARM is not valid. (may be null)
-1202	Transaction status passed to ARM is not valid.
-1203	Output Transaction ID passed to ARM is not valid. (may be null)
-1204	Output Transaction Handle passed to ARM is not valid. (may be null)
-1300	Metric ID passed to ARM is not valid.
-1301	Metric Format passed to ARM is not valid.
-1302	Metric Usage passed to ARM is not valid.
-1303	Metric Unit passed to ARM is not valid.
-1304	Output Metric ID passed to ARM is not valid. (may be null)
-1305	Metric name is not valid, must not start with ARM.
-1400	Correlator passed to ARM is not valid.
-1401	Flag number passed to ARM is zero. It is 1 (`app trace flag`) or 2 (`agent trace flag`.
-1402	Flag number passed to ARM is not valid.

Table 83. General ARM response codes (continued)

Code	Description
-1500	Charset passed in to ARM is not supported.
-1501	An error occurred that was not predefined by ARM.
-2000	ARM engine is not running.
-2001	Communication is not possible from `libarm` to the ARM engine. Communication is not possible from the ARM engine to `libarm`.
-2002	Communication is not possible from the ARM engine to `libarm`.
-2733	Invalid ID.
-50000	Transaction is not registered.
-50001	Application is not registered.
-50002	Registered application ID is not valid.
-50003	Started transaction handle is not valid.

Additional Java error codes

Table 84 describes the ARM response codes related to Java errors.

Table 84. ARM response codes for Java errors

Code	Description
-15001	Loading of the Native Library failed, `libarmjni4` not found
-15051	Size of identity names array does not match size of values array
-15052	Identity name index out of bounds
-15053	Identity name array is null
-15054	Identity value index out of bounds
-15055	Identity value array is null
-15056	Context name index out of bounds
-15057	Context name array is null
-15101	Identity Properties is not valid
-15151	`ApplicationDefinition` is null
-15152	`ApplicationDefinition` is not valid
-15153	Context value index out of bounds
-15154	Context value array is null
-15201	Application object is null
-15202	Transaction Definition is null
-15203	Context value index out of bounds
-15204	Context values array is null
-15205	Transaction context sub-buffer will not be created
-15206	Transaction is inactive
-15207	Transaction is active
-15208	Unblock called without block
-15208	Unbind called without bind
-15251	`ApplicationDefinition` is null

Table 84. ARM response codes for Java errors (continued)

Code	Description
-15252	`ApplicationDefinition` is not valid
-15253	Identity Properties is not valid
-15301	Metric Group definition index out of bounds
-15302	Metric Group definition array is null
-15351	Metric group index out of bounds
-15352	Metric group array is null
-15353	Metric group array entry is null
-15354	Metric group array is not valid
-15355	Metric group array and `MetricGroupDefinition` mismatch
-15356	`MetricGroupDefinition` is null
-15357	`MetricGroupDefinition` is not valid
-15401	`ApplicationDefinition` is null
-15402	`ApplicationDefinition` is not valid
-15451	`ArmMetricDefinition` is null
-15452	`ArmMetricDefinition` is not valid
-15501	`ArmMetricDefinitionGroup` is not valid
-15551	`ArmTransactionWithMetrics Definition` is not valid
-15601	`ArmTransactionWithMetrics Definition` is not valid
-15651	`ArmMetricString` string is too long
-15701	Internal Error, `Arm40Token` constructor array is null
-15702	Internal Error, `Arm40Token` Invalid input array
-15703	Internal Error, `Arm40Token` Invalid Offset
-15704	Internal Error, `Arm40Token` length error

Appendix H. Data Collector for WebSphere Message Broker reference

ConfigDC syntax

Use the **configDC** script to enable and configure the Data Collector for WebSphere Message Broker environment.

```
configDC.sh/bat [-enable |-disable] broker_installdir
```

where:

- `-enable`, configures the environment to load the KK3UserExit
- `-disable`, removes the KK3UserExit from the environment
- *broker_installdir*, Message Broker installation directory (optional)

  If *broker_installdir* is specified, the Data Collector for WebSphere Message Broker is enabled for that broker only.

  If *broker_installdir* is not specified, the Data Collector for WebSphere Message Broker checks for `$MQSI_WORKPATH` to determine whether to enable the current broker installation or all installations.

KK3.dc.properties

Use the `k3/config/KK3.dc.properties` configuration file to configure Data Collector for WebSphere Message Broker. The configuration file is reloaded whenever it is updated.

Apply settings globally using the prefix `default`. For example, enable the Data Collector for WebSphere Message Broker:

```
# Enable monitoring for all execution groups
default.monitor=on
```

Enable data to be sent to the Transaction Collector:

```
default.tt.enabled=true
```

Specify the address of the Transaction Collector. For example:

```
default.tt.serverstring=tcp:127.0.0.1:5455
```

Specify this directory to integrate with MQ Tracking:

```
default.ttdc.mq.installdir=path-to-MQTracking-installation-directory
```

For example, `default.ttdc.mq.installdir=C:\IBM\ITM\TMAITM6\kth` on Windows, or `/opt/IBM/WMB/aix533/th` on AIX.

Integrate with the ITCAM for SOA Services Management agent. When enabled, Data Collector for WebSphere Message Broker populates the ITCAM for SOA Services Management agent workspaces and responds to ITCAM for SOA take action commands.

```
default.kd4.enabled=true
```

When the setting has been enabled to integrate with the ITCAM for SOA agent (`default.kd4.enabled=true`), the global settings for monitor, log, and trace are ignored and are overwritten by per-Execution Group settings controlled by the

ITCAM for SOA agent. View these settings in ITCAM for SOA's Services Management Agent workspace. Modify the settings using the ITCAM for SOA take action commands.

Data Collector for WebSphere Message Broker produces an operator log with diagnostic messages. Set the logging level to `info`, `warn`, or `error`.

```
default.log=info
```

If required, Data Collector for WebSphere Message Broker can produce trace logs for L3 support. Only enable trace logs when troubleshooting problems.

```
default.trace=on
```

Appendix I. ADO.NET supported namespaces

The following table lists the ADO.NET client side APIs that are monitored by the .NET Data Collector.

Table 85. ADO.NET supported namespaces

configdc interface	.NET Namespace	Class	Method
adsi	System.DirectoryServices	DirectorySearcher	SearchResultCollection FindAll()
			SearchResult FindOne()
		DirectoryEntry	void DeleteTree()
			DirectoryEntry CopyTo(DirectoryEntry)
			DirectoryEntry CopyTo(DirectoryEntry, string)
			void MoveTo(DirectoryEntry)
			void RefreshCache()
			void RefreshCache(string[])
			void Rename(string)
ldap	System.DirectoryServices. Protocols	LdapConnection	void Bind()
			void Bind(NetworkCredential)
			void Dispose()
			void Dispose(bool)
			DirectoryResponse SendRequest(DirectoryRequest)
db2, odbc, oledb, oracle, sql	System.Data.Common	DbDataAdapter	int Fill(DataSet)
			int Fill(DataTable)
			int Fill(DataSet, string)
			int Fill(DataTable, IDbCommand, CommandBehavior)
			int Fill(int, int, DataTable[])
			int Fill(DataTable[], int, int, IDbCommand, CommandBehavior)
			int Fill(DataSet, int, int, string, IDbCommand, CommandBehavior)
			int Update(DataRow[])
			int Update(DataSet)
			int Update(DataTable)
			int Update(DataRow[], DataTableMapping)
			int Update(DataSet, string)
		DbCommand	DbDataReader ExecuteReader()

Table 85. ADO.NET supported namespaces (continued)

configdc interface	.NET Namespace	Class	Method
			DbDataReader ExecuteReader(CommandBehavior)
odbc	System.Data.Odbc	OdbcCommand	int ExecuteNonQuery()
			OdbcDataReader ExecuteReader()
			OdbcDataReader ExecuteReader(CommandBehavior)
			Object ExecuteScalar()
oleDb	System.Data.OleDb	OleDbCommand	int ExecuteNonQuery()
			OleDbDataReader ExecuteReader()
			ExecuteReader(CommandBehavior)
			Object ExecuteScalar()
oracle	System.Data.OracleClient	OracleCommand	int ExecuteNonQuery()
			int ExecuteOracleNonQuery(OracleString)
			Object ExecuteOracleScalar()
			OracleDataReader ExecuteReader()
			OracleDataReader ExecuteReader(CommandBehavior)
			Object ExecuteScalar()
sql	System.Data.SqlClient	SqlCommand	int ExecuteNonQuery()
			SqlDataReader ExecuteReader()
			SqlDataReader ExecuteReader(CommandBehavior)
			Object ExecuteScalar()
			XmlReader ExecuteXmlReader()
db2	IBM.Data.DB2	DB2Command	int ExecuteNonQuery()
			DataReader ExecutePageReader(int, int)
			Db2DataReader ExecuteReader()
			Db2DataReader ExecuteReader(CommandBehavior)
			Db2ResultSet ExecuteResultSet(Db2CursorType)
			Db2ResultSet ExecuteResultSet(Db2ResultSetOptions)
			Db2ResultSet ExecuteResultSet(CommandBehavior, DB2CursorType)
			Db2ResultSet ExecuteResultSet(CommandBehavior, DB2CursorType, bool)
			Db2ResultSet ExecuteResultSet(CommandBehavior, Db2ResultSetOptions)

Table 85. ADO.NET supported namespaces (continued)

configdc interface	.NET Namespace	Class	Method
			Db2ResultSet ExecuteResultSet(CommandBehavior, Db2ResultSetOptions, bool)
			DB2Record ExecuteRow()
			Object ExecuteScalar()
			XmlReader ExecuteXmlReader()

Appendix J. 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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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.

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 multiple Data Collector plug-ins and other monitors and computes aggregates for use by the Transaction Reporter. Aggregation agents include the Transaction Collector and Web Response Time 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 plug-ins 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 plug-ins.

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

Special characters

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Printed in USA

SC14-7408-01