webMethods EntireX

Application Monitoring

Version 10.8

October 2022

WEBMETHODS

This document applies to webMethods EntireX Version 10.8 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: EXX-APPMON-108-20220601

Table of Contents

Preface

This documentation explains how to receive response-time data from your distributed applications. It is organized under the following headings:

Introduction	What is application monitoring? Sample scenarios in which the EntireX Broker and the EntireX Adapter are used.
Components	List of components that can be used to monitor distributed application scenarios.
Setting Up	How to set up EntireX Broker and the EntireX Adapter. Information on the configuration file for the Application Monitoring Data Collector. How to start and stop the Application Monitoring Data Collector.
KPI Definitions	Describes the key performance indicators (KPIs) monitored by the Application Monitoring Data Collector.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction to Application Monitoring

Scope

Application Monitoring is an EntireX feature that enables you to monitor the response times in your distributed applications, and it also enables you to monitor certain error situations. The heart of Application Monitoring is the EntireX Application Monitoring Data Collector, which collects the response time data of each involved software component of selected synchronous EntireX RPC services. The Application Monitoring Data Collector stores the KPI values in CSV (comma-separated values) files. The files can be processed by any third-party tool that supports CSV files, for example Microsoft Excel. Alternatively, you can hook in your own monitoring back end, using the callback user exit of the Data Collector.

When Application Monitoring is enabled, each call to the service by a client application is monitored. You can measure

- overall service response times
- network transport times
- EntireX Broker processing
- waiting times
- RPC server processing times

You can also measure

- time spent for database calls (if Natural and mainframe Adabas are installed on the back end)
- time spent for DB2 calls (if applicable and depending on the platform)

Each Software AG enterprise product involved concatenates the monitored time(s) with the service call. When the call returns to the client, the client RPC runtime provides the event data to the Application Monitoring Data Collector.

Scenarios with webMethods Integration Server

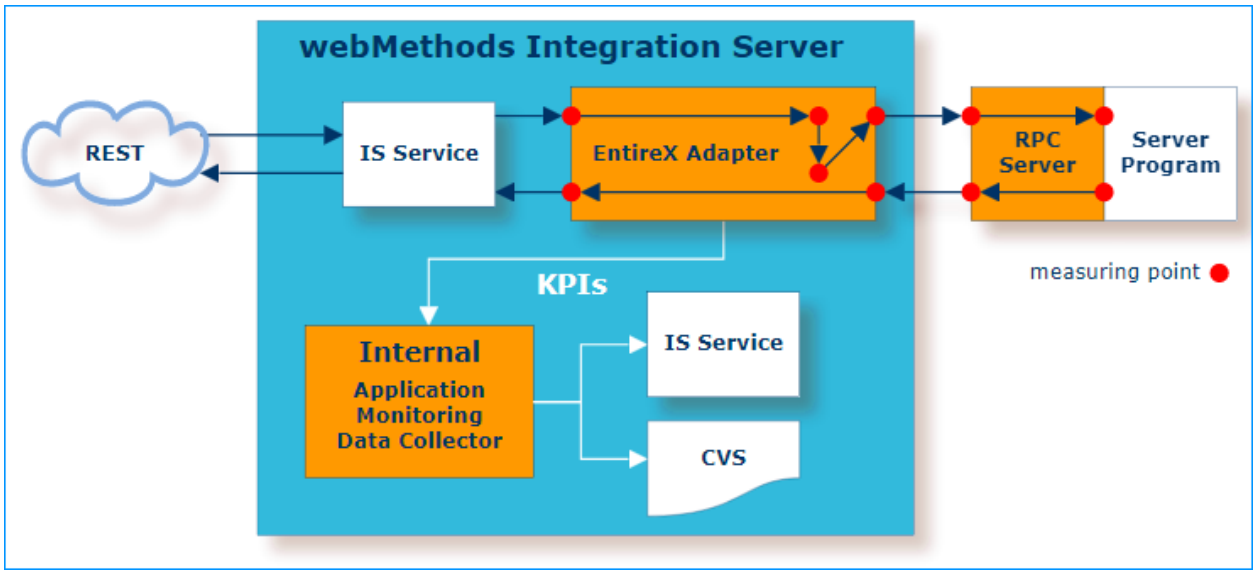
For webMethods Integration Server scenarios, the Application Monitoring Data Collector is available inside the Integration Server. This section describes the following scenarios:

- REST Client or IS Service Calls RPC Server
- REST Client or IS Service Calls CICS ECI, IMS Connect or AS/400
- REST Client or IS Service Calls CICS Socket Listener

- RPC Client calls IS Service or REST API

REST Client or IS Service Calls RPC Server

In this scenario, REST clients call into an RPC server via an IS service: inbound calls from the viewpoint of the webMethods Integration Server, or RPC server. The following diagram shows the KPI measuring points and flow when a REST client or IS service calls an RPC server.

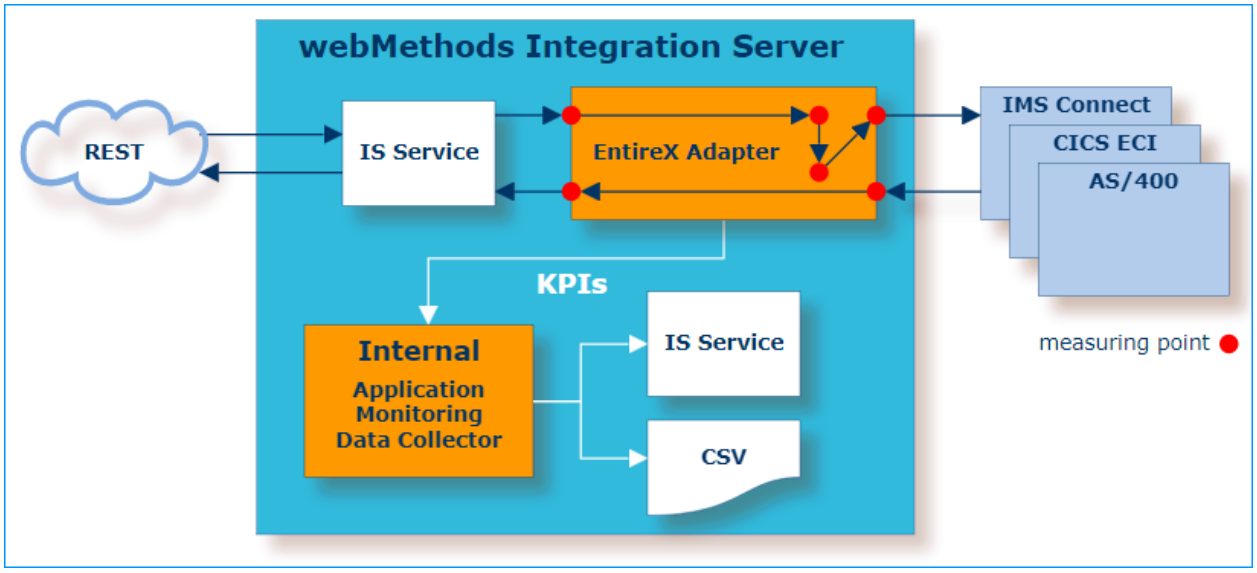


The KPI measuring points comprise the Integration Server and the RPC server.

REST Client or IS Service Calls CICS ECI, IMS Connect or AS/400

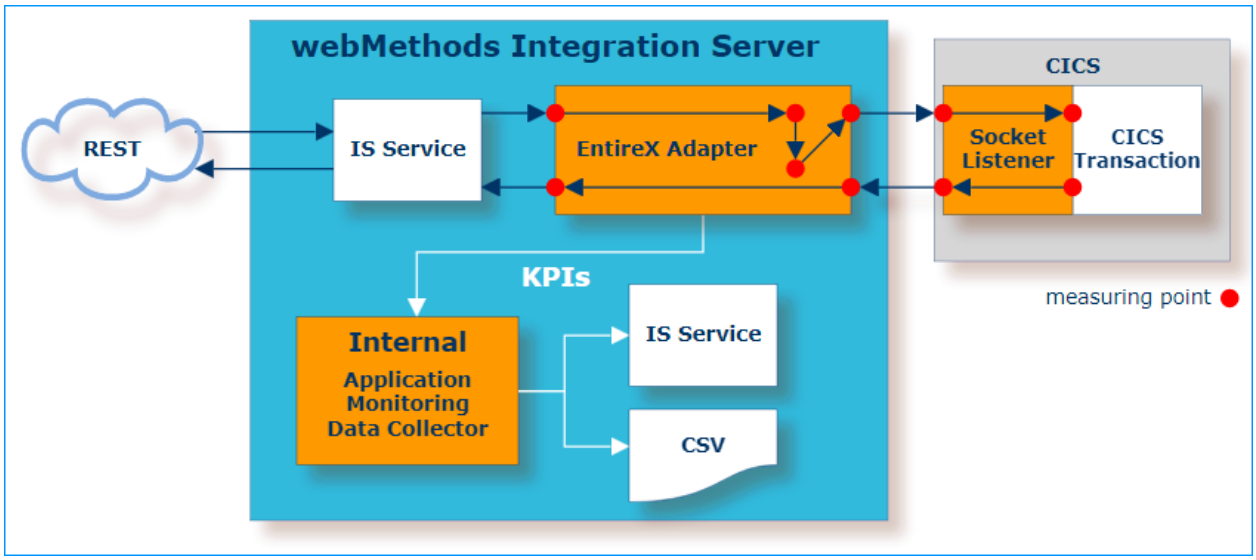
In this scenario, REST clients call into the back end (CICS, IMS or AS/400) via an IS service: inbound calls from the viewpoint of the webMethods Integration Server, or back end. The following diagram shows the KPI measuring points and flow when a REST client or IS service calls the back end using a CICS ECI Connection, IMS Connect Connection or AS/400 Connection.

You can monitor the sum of the transport time to CICS ECI, IMS Connect or AS/400 plus the time spent in the CICS transaction, IMS transaction or AS/400 program.



The KPI measuring points comprise the Integration Server.

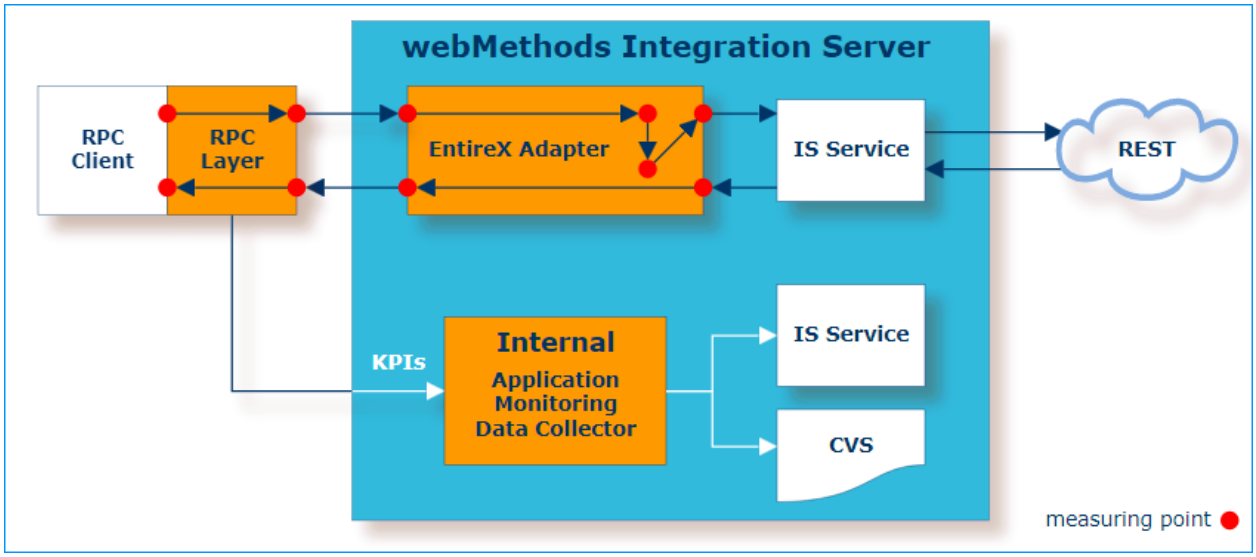
REST Client or IS Service Calls CICS Socket Listener



The KPI measuring points comprise the Integration Server and Socket Listener.

RPC Client calls IS Service or REST API

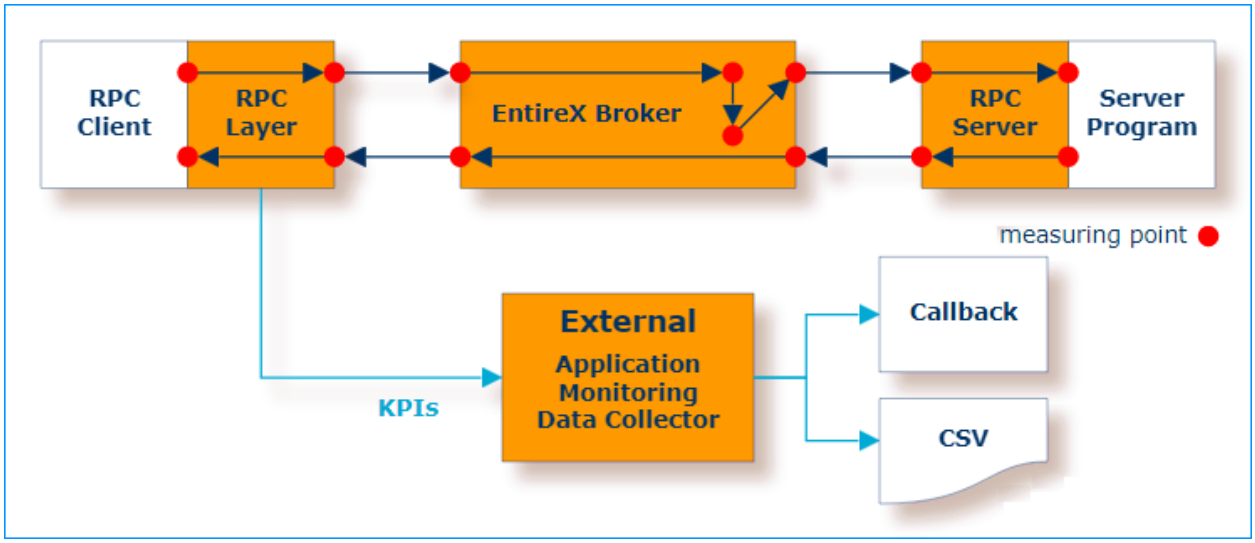
This scenario illustrates the opposite direction: outbound calls of an RPC client calling a REST service via an IS service. The following diagram shows the KPI measuring points and flow when an RPC client calls a webMethods IS service or REST API.



The KPI measuring points comprise the RPC client and Integration Server.

Scenarios with EntireX Broker

There are also scenarios that do not use webMethods Integration Server, for example calls to/from web services or another RPC client through RPC server communication. Where EntireX Broker is the communication hub, the Application Monitoring Data Collector runs as a separate standalone component, often called the *External* Application Monitoring Data Collector.



RPC clients support Application Monitoring without additional configuration, including the EntireX Adapter with an EntireX RPC connection, which is technically an RPC client. So does an RPC server, including the EntireX Adapter with an EntireX RPC Listener connection, which is technically an RPC server.

Processing KPI Data with EntireX Adapter

You can process your KPI data using the Application Monitoring Data Collector or an Integration Server service:

- Processing KPI Data with CSV Files

Processing KPI Data with CSV Files

The Application Monitoring Data Collector collects the response time data of each involved software component of selected synchronous EntireX RPC services and stores the KPI (key performance indicator) values in CSV (comma-separated values) files. The files can be processed by any third-party tool that supports CSV files. For webMethods Integration Server scenarios, the Data Collector is available inside the EntireX Adapter.

See *Application Monitoring* in the EntireX Adapter documentation for configuration details.

Processing KPI Data with Integration Server Service

As an alternative to storing the KPIs in CSV files, you can invoke an IS service. The monitoring KPIs can be consumed by an Integration Server service, and the monitoring data can be further processed.

Processing KPI Data with External Application Monitoring Data Collector

The External Data Collector runs as a pre-started service controlled by a daemon, for example the `cron` daemon in a UNIX system, or a Windows service under the Windows operating system.

For more information see ***Setting up the External Application Monitoring Data Collector*** under *Setting up Application Monitoring*.

With the External Application Monitoring Data Collector you can process your KPIs with any tool that supports CSV files or you can use the callback user exit of the External Data Collector:

- Processing KPI Data with CSV Files
- Processing KPI Data with Callback User Exit

Processing KPI Data with CSV Files

The External Application Monitoring Data Collector collects the response time data of each involved software component of selected synchronous EntireX RPC services and stores the KPI (key performance indicator) values in CSV (comma-separated values) files. The files can be processed by any third-party tool that supports CSV files.

Processing KPI Data with Callback User Exit

As an alternative to storing the KPIs in CSV files, you can use the callback user exit of the External Application Monitoring Data Collector. Use this approach if you want to feed arbitrary monitoring back ends in real time.

To use the callback user exit

1 Write a Java class that implements the `DataCollectorCallback` interface.

2 Make it known to the External Application Monitoring Data Collector.

For more information see ***Callback User Exit*** under *Setting up Application Monitoring*.

3 Components that Support Application Monitoring

You can monitor distributed application scenarios for transport methods TCP/IP or SSL that make use of the following components:

	Component		z/OS	UNIX	Windows	z/VSE	BS2000	Comment
	EntireX Broker	x	x	x	x	x	See *Dynamic Configuration and Setup for EntireX Broker*.	
	EntireX Adapter		x	x			See *Dynamic Configuration and Setup for EntireX Adapter*.	
Clients	Listener for XML/SOAP	x	x	x				
	Java RPC Client	x	x	x				
	.NET RPC Client			x				
	Natural RPC Client	x (1)	x	x	x			
	COBOL RPC Client	x			x			
	C RPC Client		x	x				
Servers	EntireX RPC Servers	x	x	x			All EntireX RPC servers support Application Monitoring. See *EntireX RPC Servers*.	
	Natural RPC Server	x (1)	x	x				

Notes:

1. For z/OS, make sure the EXX load library is part of your steplib chain. We recommend using stub `NATETB23` for all of your Natural RPC environments and a Natural configuration allowing a dynamic load of the stub. This can be achieved by using the following Natural parameters:

```
RCA=(BROKER) RCALIAS=(BROKER,NATETB23)
```

If your broker stub is statically included, you will need to relink your Natural nucleus.

4 Setting up Application Monitoring

General Information

EntireX Broker and/or the EntireX Adapter serve as the central components which control the data flow. Their configuration defines the following:

- whether application monitoring is enabled or disabled
- the services that are used for monitoring (only for EntireX Broker), see *Fine-tuning Application Monitoring for Selected Services*
- the Application Monitoring Data Collector to which the measuring data is sent.

RPC client and RPC server applications (see *Components that Support Application Monitoring*) automatically support Application Monitoring without additional setup. This includes EntireX RPC Connection and EntireX RPC Listener Connection of the EntireX Adapter, which are used together with an EntireX Broker as the central component. See *Dynamic Configuration and Setup for EntireX Adapter*.

Configuration and Setup

The Application Monitoring Data Collector inside the EntireX Adapter and the External Application Monitoring Data Collector are interchangeable. You can use the Data Collector inside the EntireX Adapter for scenarios with the EntireX Broker and equally, the External Application Monitoring Data Collector for scenarios with the webMethods Integration Server.

Whatever architecture you choose, you can configure Application Monitoring dynamically while the EntireX Adapter or EntireX Broker is running. The change is effective immediately, and you do not need to restart the EntireX Adapter or EntireX Broker. This section covers the following topics:

Dynamic Configuration and Setup for EntireX Adapter

This setup applies to the following connection types:

- EntireX Direct RPC Connection
- EntireX Direct RPC Listener Connection
- IMS Connect Connection
- CICS ECI Connection
- CICS Socket Listener Connection

- AS/400 Connection

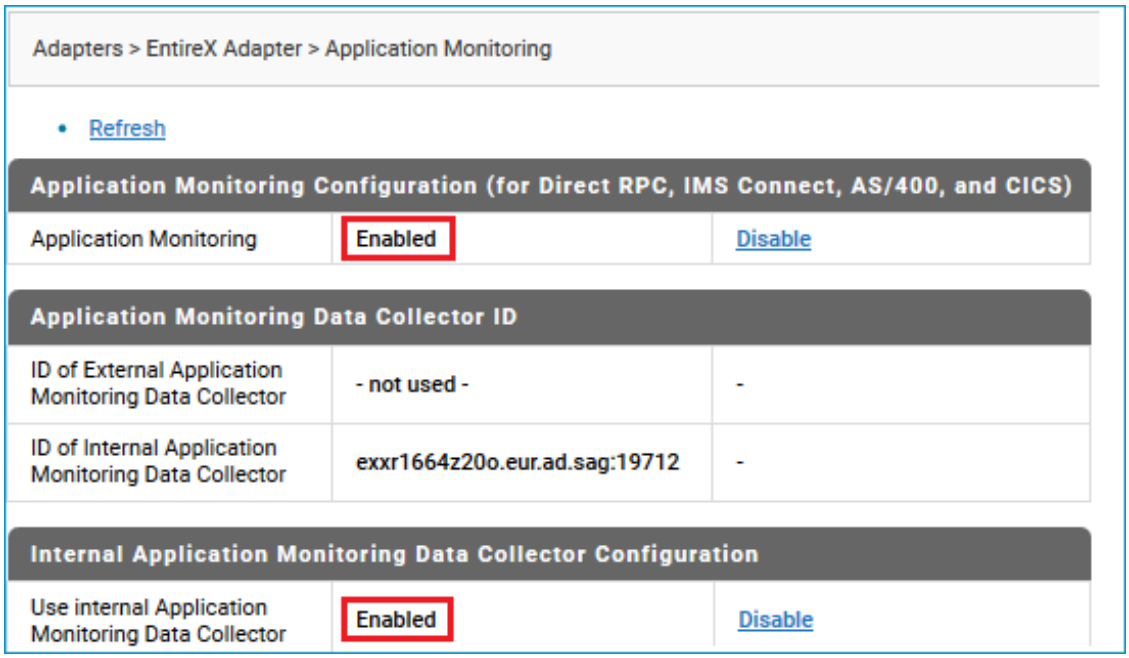
Note: If you are using an EntireX RPC Connection or EntireX RPC Listener Connection, you need to enable Application Monitoring in the EntireX Broker to which the connection is made. See *Dynamic Configuration and Setup for EntireX Broker*.

You can choose an Internal Application Monitoring Data Collector (recommended) or an External Collector.

- **Internal Collector**

To configure the Internal Application Monitoring Data Collector in EntireX Adapter dynamically

1. In the webMethods IS Administration GUI, choose **Adapters** > **EntireX Adapter** > **Application Monitoring**.
2. Enable **Application Monitoring**.
3. Enable **Use internal Application Monitoring Data Collector**.



The collector runs as a server in the Direct RPC component and is started and stopped automatically when the Direct RPC component is started or stopped.

- **External Collector**
  Alternatively, the monitoring KPIs can be routed to an External Application Monitoring Data Collector, using either
  - a collector running internally within Direct RPC in a different instance of Integration Server, or

- a collector pre-started as a service, as described under *Setting up the External Application Monitoring Data Collector*

Dynamic Configuration and Setup for EntireX Broker

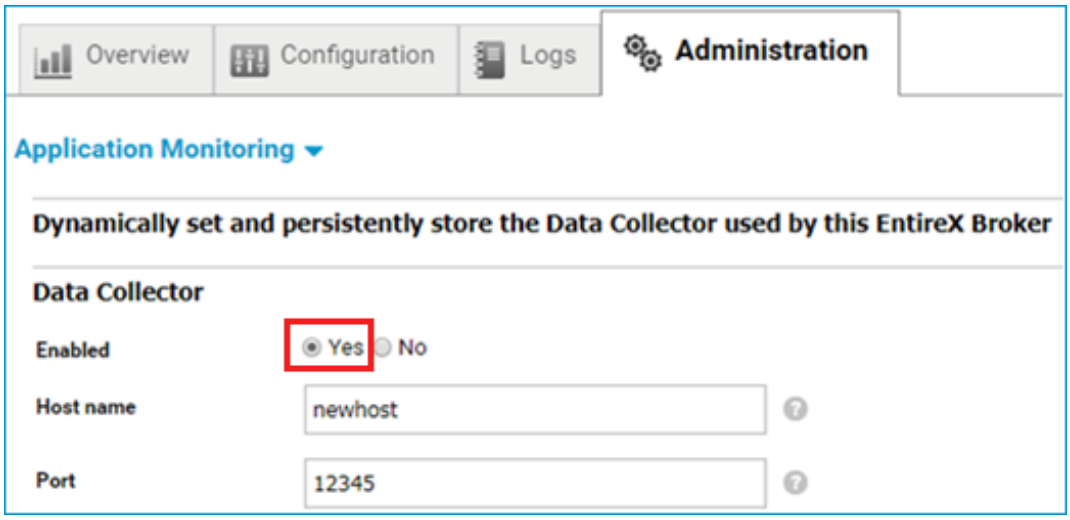
Configuring the External Application Monitoring Data Collector in EntireX Broker scenarios depends on the platform the broker is running on. You can also fine-tune Application Monitoring for selected services. This section covers the following topics:

- Configuration and Setup under UNIX and Windows
- Configuration and Setup under z/OS
- Fine-tuning Application Monitoring for Selected Services

Configuration and Setup under UNIX and Windows

Use the Command Central GUI or command-line interface:

- **GUI**
  Choose **Administration** > **Application Monitoring** and enable **Data Collector**.



  See *Changing the Application Monitoring Settings Dynamically* under *Administering EntireX Broker using the Command Central GUI*.

- **Command Line**
  See *Changing the Application Monitoring Settings Dynamically* under *Administering EntireX Broker using the Command Central Command Line*.

Configuration and Setup under z/OS

Define the endpoint for the External Data Collector in the broker attribute file, see *Application Monitoring-specific Attributes* in the platform-independent Administration documentation. You can then enable and disable application monitoring dynamically, see *Application Monitoring-specific Commands* in the z/OS Administration documentation. To permanently enable application monitoring, set `APPLICATION-MONITORING=YES` under *Broker-specific Broker Attributes*.

Fine-tuning Application Monitoring for Selected Services

Using specific attributes in the broker attribute file, the broker can be configured to enable application monitoring for selected services. Configuration example:

```
DEFAULTS = BROKER
  APPLICATION-MONITORING = YES

DEFAULTS = APPLICATION-MONITORING
  COLLECTOR-BROKER-ID = server12:57900

DEFAULTS = SERVICE
  APPLICATION-MONITORING-NAME = Payroll_Application
  CLASS = RPC, SERVER = HR, SERVICE = CALLNAT, APPLICATION-MONITORING = YES, ↩
APPLICATION-MONITORING-NAME = HR_Application
  CLASS = RPC, SERVER = *, SERVICE = CALLNAT, APPLICATION-MONITORING = YES
```

With this example configuration, application monitoring is enabled for all RPC/*/CALLNAT services. The service RPC/HR/CALLNAT uses the application monitoring name "HR_Application", all other services use the name "Payroll_Application". The Application Monitoring Data Collector runs on a host with the name "server12" and uses the port 57900.

Notes:

1. There are broker-specific and service-specific attributes for application monitoring, and there are also application monitoring-specific attributes. For detailed information, see *Broker Attributes*.
2. Changes in the broker attribute file regarding fine-tuning application monitoring require a restart of
   - the broker if the automatic update mode of the broker is switched off, see `SERVICE-UPDATES=NO` under *Broker-specific Attributes* in the Platform-independent Administration documentation
   - all involved RPC servers in any case.

Setting up the External Application Monitoring Data Collector

This section covers the following topics:

Configuring the Application Monitoring Data Collector

The configuration file *entirex.appmondc.properties* controls the startup of the Application Monitoring Data Collector. It is located in the *config* directory of your EntireX installation.

As a rule, it is not necessary to change the settings in this file after the installation. However, if required, you can change the following parameters:

Parameter	Description
`entirex.appmondc.port`	The TCP/IP port on which the Application Monitoring Data Collector accepts the monitoring data. This value is set during the installation of the Application Monitoring Data Collector.
`entirex.appmondc.directory`	The name of the directory which will contain the CSV data files. The default value is *<EntireX-install-dir>/appmondc/*. A data file has the name *appmon<YYYYMMDD>.<HHMMSS>.csv*. In addition, an overview file with the name *appmon.overview.v1.csv* is created.

`entirex.appmondc.loglevel`	The log level for the log files. Possible values are:
	
	`OFF`
	`FATAL`
	`ERROR`
	`WARNING`
	`INFO`
	`DEBUG`
	`TRACE`
	
	The default value is `ERROR`.
	
	Log files are always stored in the directory *<EntireX-install-dir>/appmondc/*. This is independent of the setting of the `entirex.appmondc.directory` parameter.

`entirex.appmondc.maxlines`	The maximum number of rows per CSV data file. If the limit is reached, a new file is created. The default value is 100000.

Parameter	Description
`entirex.appmondc.filesperday`	Automatically create a new CVS data file every day. The default value is `no`.
`entirex.appmondc.usezeroasnullvalue`	Use "0" instead of an empty entry as the null value for all numeric KPI values in the CSV file. The default value is `no`.
`entirex.appmondc.callback.class`	Java class name for the callback user exit.
`entirex.appmondc.callback.classpath`	Classpath name in URL notation for the callback user exit.

Starting and Stopping the Application Monitoring Data Collector

This section covers the following topics:

- UNIX
- Windows

UNIX

The scripts mentioned below are located in the *bin* directory of your EntireX installation. By default, this is */opt/softwareag/entirex/bin*.

To start the Application Monitoring Data Collector

- Run the script *appmondc.bsh* from a shell.

To stop the Application Monitoring Data Collector

- Run the script *stopappmondc.bsh* from a shell.

Windows

The Application Monitoring Data Collector is installed as an application and as a Windows service. During the installation, you can specify that the Windows service is to be started automatically. The name of this service is "Software AG EntireX Application Monitoring Data Collector 10.8".

Note: You can access the list of services by opening the Start menu and then entering "services.msc" in the search box. There, you can start and stop the service manually.

To start the Application Monitoring Data Collector as an application

- Choose the following from the Windows Start menu:

  Programs > Software AG > Start Servers > Start EntireX Application Monitoring Data Collector 10.8

Or:

Run the script *appmondc.bat* which is located in the *bin* directory of your EntireX installation.

To stop the Application Monitoring Data Collector (service and application)

- Choose the following from the Windows Start menu:

  Programs > Software AG > Stop Servers > Stop EntireX Application Monitoring Data Collector 10.8

  Or:

  Run the script *stopappmondc.bat* which is located in the *bin* directory of your EntireX installation.

Callback User Exit

The Application Monitoring Data Collector provides a callback functionality for the processing of incoming events. A user exit can be specified to implement the callback. Whenever the Data Collector receives a monitoring event, the KPI values are writen to the CSV file and the callback is invoked. The KPI values of the event are passed to the callback. Possible use cases are:

- trigger an action based on specific KPI values (e.g. the KPIs indicate a failed request)
- write the KPI values to another data store or in a format that is different from a CSV file

To enable the user exit callback, use the property `entirex.appmondc.callback.class` to specify the class name of the user exit implementation. The class is loaded using the standard classpath. You can specify a separate classpath with the property `entirex.appmondc.callback.classpath`. Note that for the classpath, a file or HTTP URL must be specified (on Windows replace the "\" character with "/"). Your user exit class must implement the Java interface `com.softwareag.entirex.appmondc.DataCollectorCallback`. This Java interface has the following methods:

```
/**
 * Initialize the callback handler.
 * @param directory The name of the directory that will contain the CSV data files ↩
(set by property "entirex.appmondc.directory")
 *
 * @throws Exception
 */
public void start(String directory) throws Exception;

/**
 * Stop the callback handler.
 */
public void stop();

/**
```

```
 * Process an event. This corresponds to an entry in the CSV file.
 * The map contains all KPIs which have a value.
 * The key names of the map are identical to the KPI names.
 *
 * @param kpis The KPI map.
 * @throws Exception
 */
public void processEvent(Map<String, String> kpis) throws Exception;
```

5 KPI Definitions for Application Monitoring

General Information

The tables below describe the KPIs (key performance indicators) monitored by the Application Monitoring Data Collector. The following scenarios are supported. Note that Adapter scenarios require the corresponding connection type.

Scenario	Supported by		
	EntireX Broker	EntireX Adapter...	...and Connection Type
RPC	x	x	Direct RPC
CICS ECI		x	CICS ECI
CICS Socket Listener		x	CICS Socket Listener
IMS Connect		x	IMS Connect
AS/400		x	AS/400

Each scenario has a different set of KPIs for successful requests and for failed requests. For a successful request, the KPI "ErrorCode" is always empty. For a failed request this KPI always has a value.

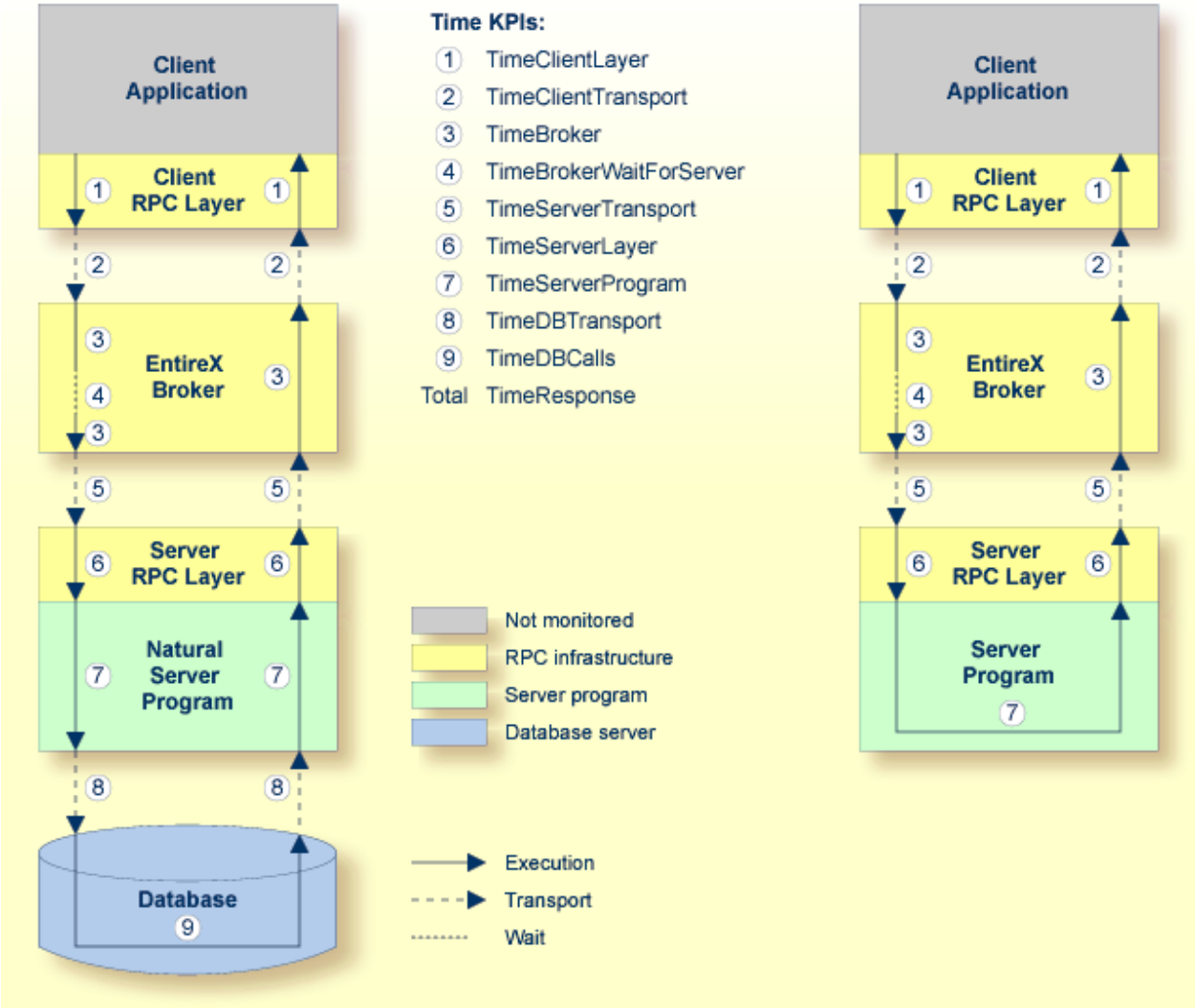
Each KPI is represented as a row in the CSV file produced by the data collector. The KPI name is identical to the row name in the first column of the CSV file. There is only one common layout of the CSV file. Therefore, depending on the scenario, KPIs may have no values. In this case, the column has an empty entry in the corresponding row. This can be changed to the value "0" by setting the parameter `entirex.appmondc.usezeroasnullvalue` in the configuration file *entirex.appmondc.properties* (see ***Configuring the Application Monitoring Data Collector***) or by changing the coresponding parameter in the configuration of the EntireX Adapter. See `Use "0" as the null value for numeric KPI values` in the EntireX Adapter documentation.

If the KPIs are consumed by an Integration Server service then each KPI is represented by an input parameter of the service using the same name. If a KPI has no value, then the corresponding input parameter has no value.

The sequence of the KPIs in the tables below is the same as the sequence of the KPIs in the CSV file.

Detailed Illustration of Response Time KPIs

The following graphic shows the detailed meanings of the response time KPIs that are collected by the Application Monitoring Data Collector. As an example, a remote procedure call (RPC) from a client application to a server program is illustrated. For a Natural server program, database calls can be monitored too; this is illustrated on the left side of the graphic. Note that the database transport time (**TimeDBTransport** KPI) is only available for a database call against an Adabas server. For other databases, the database transport time is included in the database calls time (**TimeDBCalls** KPI). The right side illustrates a non-Natural server program where database calls cannot be monitored.



1. The client application issues a remote procedure call and the client RPC layer gets control. At the end of a transaction, the client RPC layer gives the control back to the client application. The time spent in the client RPC layer is monitored by the **TimeClientLayer** KPI.
2. The client RPC layer calls the EntireX Broker. The transport time between the client RPC layer and the EntireX Broker plus the transport time spent on the way back is monitored by the **TimeClientTransport** KPI.
3. The time spent in the EntireX Broker is monitored by the **TimeBroker** KPI. The KPI value does not include the amount of time that the EntireX Broker spends waiting for an available server.
4. The amount of time that the EntireX Broker spends waiting for an available server is monitored by the **TimeBrokerWaitForServer** KPI.
5. When the EntireX Broker calls the server program, the server RPC layer receives the call first. The transport time between the EntireX Broker and the server RPC layer plus the transport time spent on the way back is monitored by the **TimeServerTransport** KPI.
6. The time spent in the server RPC layer is monitored by the **TimeServerLayer** KPI.
7. The server RPC layer forwards control to the Natural server program. The time spent in the Natural server program is monitored by the **TimeServerProgram** KPI. The KPI value does not include the time spent for database calls.
8. The Natural server program calls a database. The transport time between the Natural server program and the database plus the transport time spent on the way back is monitored by the **TimeDBTransport** KPI. This KPI is only available for Natural RPC servers issuing database calls against an Adabas server.
9. The time spent for database calls is monitored by the **TimeDBCalls** KPI. For non-Adabas databases, the KPI value includes also the transport time required to reach the database server. This KPI is only available for Natural RPC servers.

- The **TimeResponse** KPI reflects the complete response time on the round trip from the client to the server. It is therefore the sum of the KPIs mentioned above.

KPIs to Collect Response Time, Statistical Information etc.

Use the following KPIs to collect response times and find bottlenecks in distributed scenarios. Collect statistical information on how often a service is used.

- KPIs for RPC - Successful Requests
- KPIs for CICS ECI - Successful Requests
- KPIs for CICS Socket Listener - Successful Requests
- KPIs for IMS Connect - Successful Requests

- KPIs for AS/400 - Successful Requests

KPIs for RPC - Successful Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "RPC".
ApplicationName	"*application-name*" as defined by the broker attribute `APPMON-NAME`. If the broker attribute is not specified, the server address is used; for example `RPC/SRV/CALLNAT`.
Address	The broker ID and the server address of the RPC request.
TimeResponse	The complete response time (roundtrip from client to server and back) in microseconds.
TimeClientLayer	The time spent in the client RPC layer in microseconds.
TimeClientTransport	The transport time from the client to the broker and back in microseconds.
TimeBroker	The time spent in the broker (active processing) in microseconds.
TimeBrokerWaitForServer	The time spent in the broker waiting for an available server in microseconds.
TimeServerTransport	The transport time from the broker to the server and back in microseconds.
TimeServerLayer	The time spent in the server RPC layer (runtime and stub) in microseconds.
TimeServerProgram	The time spent in the user program (called by the RPC server) in microseconds. For Natural programs on a mainframe, this time does not include the database times. For other programs, the database times are included.
TimeDBCalls	The time spent for database calls in microseconds. For an Adabas database, this is the time the Adabas server needs to process the database call (“client wait time”). For other databases, the DB calls time includes also the DB transport time. [1]
TimeDBTransport	The transport time from the Natural user program to the Adabas router and back including the client receiving time in microseconds. [1, 2]
Program	The program name.
ClientApplication	The client application name as defined in the broker control block.
ClientHost	The client host name.
ClientUser	The client user ID.
LengthRequest	The length of the RPC request in bytes.
LengthReply	The length of the RPC reply in bytes.
LengthTotal	The total length of the RPC call (request plus reply) in bytes.
DBCalls	The number of database calls (including system file calls, without Natural Security calls). [1]

KPI Name	Description
ErrorCode	Always empty.
MessageID	The message ID of the RPC request (if available).
CorrelationID	The message ID of the RPC reply (if available).

Notes:

[1] This KPI is only available if the call is issued by a Natural RPC server on a mainframe.

[2] This KPI is only available for a database call against an Adabas server.

KPIs for CICS ECI - Successful Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "CICS ECI".
ApplicationName	"*host-name*:*port-number*" of the CICS ECI installation.
Address	The name of the Integration Server adapter service which calls CICS ECI.
TimeResponse	The complete response time of the CICS ECI request in microseconds.
TimeClientLayer	The time spent in the EntireX Adapter in microseconds.
TimeServerLayer	The sum of the transport time to CICS ECI and the time spent in the CICS user program in microseconds.
Program	The CICS program name.
ClientHost	The client host name.
ClientUser	The client user ID.
LengthRequest	The length of the CICS request in bytes.
LengthReply	The length of the CICS reply in bytes.
LengthTotal	The total length of the CICS call (request plus reply) in bytes.
ErrorCode	Always empty.
MessageID	The message ID of the CICS request (if available).

KPIs for CICS Socket Listener - Successful Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "CICS Socket Listener ".
ApplicationName	"*host-name:port-number*" of the CICS Socket Listener.
Address	The name of the Integration Server adapter service which calls the CICS Socket Listener.
TimeResponse	The complete response time of the CICS Socket Listener request in microseconds.
TimeClientLayer	The time spent in the EntireX Adapter in microseconds.
TimeServerTransport	The transport time from the adapter to the CICS Socket Listener and back in microseconds.
TimeServerLayer	The time spent in the CICS Socket Listener layer in microseconds.
TimeServerProgram	The time spent in the CICS user program (called by the CICS Socket Listener) in microseconds.
Program	The CICS program name.
ClientHost	The client host name.
ClientUser	The client user ID.
LengthRequest	The length of the CICS request in bytes.
LengthReply	The length of the CICS reply in bytes.
LengthTotal	The total length of the CICS call (request plus reply) in bytes.
ErrorCode	Always empty.
MessageID	The message ID of the CICS request (if available).

KPIs for IMS Connect - Successful Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "IMS Connect".
ApplicationName	"*host-name*:*port-number*/*datastore*" of the IMS Connect installation.
Address	The name of the Integration Server adapter service which calls IMS Connect.
TimeResponse	The complete response time of the IMS request in microseconds.

KPI Name	Description
TimeClientLayer	The time spent in the EntireX Adapter in microseconds.
TimeServerLayer	The sum of the transport time to IMS Connect and the time spent in IMS Connect, IMS and the IMS user program in microseconds.
Program	The IMS transaction name.
ClientHost	The client host name.
ClientUser	The client user ID.
LengthRequest	The length of the IMS request in bytes.
LengthReply	The length of the IMS reply in bytes.
LengthTotal	The total length of the IMS call (request plus reply) in bytes.
ErrorCode	Always empty.
MessageID	The message ID of the IMS request (if available).

KPIs for AS/400 - Successful Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "AS400".
ApplicationName	"*host-name*" of the AS/400 host.
Address	The name of the Integration Server adapter service which calls AS/400.
TimeResponse	The complete response time of the AS/400 request in microseconds.
TimeClientLayer	The time spent in the EntireX Adapter in microseconds.
TimeServerLayer	The sum of the transport time to AS/400 and the time spent in AS/400 and the AS/400 user program in microseconds.
Program	The AS/400 prorgam name.
ClientHost	The client host name.
ClientUser	The client user ID.
LengthRequest	The length of the AS/400 request in bytes.
LengthReply	The length of the AS/400 reply in bytes.
LengthTotal	The total length of the AS/400 call (request plus reply) in bytes.
ErrorCode	Always empty.
MessageID	The message ID of the AS/400 request (if available).

KPIs to Collect Error Situations

Use the following KPIs to analyze error situations. Find out which problems occur most often and need to be solved first to reach maximum stability.

KPIs for RPC - Failed Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "RPC".
ApplicationName	"*application-name*" as defined by the broker attribute `APPMON-NAME`. If the broker attribute is not specified, the server address is used; for example `RPC/SRV/CALLNAT`.
Address	The broker ID and the server address of the RPC request.
TimeResponse	The response time of the failed RPC request in microseconds.
Program	The program name.
ClientApplication	The client application name as defined in the broker control block.
ClientHost	The client host name.
ClientUser	The client user ID.
ErrorCode	The 8-digit error code (error class and number).
ErrorMessage	The error message.
MessageID	The message ID of the RPC request (if available).
CorrelationID	The message ID of the RPC reply (if available).

KPIs for CICS ECI - Failed Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "CICS ECI".
ApplicationName	"*host-name*:*port-number*" of the CICS ECI installation.
Address	The name of the Integration Server adapter service which calls CICS ECI.
TimeResponse	The response time of the failed CICS ECI request in microseconds.
Program	The CICS transaction name.
ClientHost	The client host name.
ClientUser	The client user ID.
ErrorCode	The 8-digit error code (error class and number).
ErrorMessage	The error message.
MessageID	The message ID of the CICS request (if available).

KPIs for CICS Socket Listener - Failed Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "CICS Socket Listener ".
ApplicationName	"*host-name:port-number*" of the CICS Socket Listener.
Address	The name of the Integration Server adapter service which calls the CICS Socket Listener.
TimeResponse	The response time of the failed CICS Socket Listener request in microseconds.
Program	The CICS program name.
ClientHost	The client host name.
ClientUser	The client user ID.
ErrorCode	The 8-digit error code (error class and number).
ErrorMessage	The error message.
MessageID	The message ID of the CICS request (if available).

KPIs for IMS Connect - Failed Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "IMS Connect".
ApplicationName	"*host-name*:*port-number*/*datastore*" of the IMS Connect installation.
Address	The name of the Integration Server adapter service which calls IMS Connect.
TimeResponse	The response time of the failed IMS request in microseconds.
Program	The IMS transaction name.
ClientHost	The client host name.
ClientUser	The client user ID.
ErrorCode	The 8-digit error code (error class and number).
ErrorMessage	The error message.
MessageID	The message ID of the IMS Connect request (if available).

KPIs for AS/400 - Failed Requests

KPI Name	Description
Time	The time the event has been processed by the data collector in the format "YYYY-MM-DD HH:MM:SS.SSS" using the current time zone.
Timestamp	The time the event has been processed by the data collector as a number. The number is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Scenario	The scenario identifier "AS400".
ApplicationName	"*host-name*" of the IMS Connect installation.
Address	The name of the Integration Server adapter service which calls AS/400.
TimeResponse	The response time of the failed AS/400 request in microseconds.
Program	The AS/400 transaction name.
ClientHost	The client host name.
ClientUser	The client user ID.
ErrorCode	The 8-digit error code (error class and number).
ErrorMessage	The error message.
MessageID	The message ID of the AS/400 request (if available).